



教育部高等学校自动化专业教学指导分委员会牵头规划



普通高等教育“十一五”国家级规划教材

MATLAB Solutions to Mathematical Problems in Control

控制数学问题的MATLAB求解

薛定宇 陈阳泉 著

Xue Dingyu Chen Yangquan

张庆灵 主审

Zhang Qingling

清华大学出版社

本书特色

以MATLAB语言为主线，系统地介绍控制中数学问题的求解：

- 微积分与积分变换问题求解
- 线性代数、微分方程与差分方程问题求解
- 代数方程、最优化与最优控制问题求解
- 智能控制、鲁棒控制、分数阶控制等问题求解

全新的框架、深入浅出的介绍、全部可重复的演示实例

作者简介



薛定宇 获得自动化专业学士(沈阳工业大学1985)、硕士(东北工学院1988)和博士学位(英国Sussex大学1992)，现任东北大学信息科学与工程学院教授，博士生导师。长期从事MATLAB语言、控制系统CAD等领域的教学与研究工作，相关著作被数万篇博士、硕士论文引用。



陈阳泉 获得自动化专业学士(北京钢铁学院1985)、硕士(北京工业学院1989)和博士学位(新加坡南洋理工大学1998)，现任美国尤他州立大学助理教授，自组织与智能系统中心主任、IEEE高级会员。长期从事智能控制等领域的教学与研究工作，著有学术论文200余篇，美国专利13项。


ISBN 978-7-302-15297-2

9 787302 152972 >

定价：45.00元

全国高等学校自动化专业系列教材

教育部高等学校自动化专业教学指导分委员会牵头规划



普通高等教育“十一五”国家级规划教材

MATLAB Solutions to Mathematical Problems in Control

控制数学问题的MATLAB求解

东北大学　薛定宇 Xue Dingyu

美国尤他州立大学　陈阳泉 Chen Yangquan　著

东北大学　张庆灵 Zhang Qingling　主审

清华大学出版社

北京

内容简介

控制学科中的很多内容和数学问题是密不可分的。本书系统地介绍了国际控制界最流行的MATLAB语言在数学各个分支中的应用，包括微积分与积分变换问题的求解、线性代数问题的求解、微分方程问题的求解和最优化问题的求解，并基于这些基本内容探讨了离散系统、智能控制、鲁棒控制等控制分支数学问题的求解方法，还引入了分数阶微积分问题及其在控制应用方面的新思路。

本书可作为控制学科高年级本科生和研究生的教材和主要参考书，适合开设全新的课程，也可供相关专业的研究人员参考。

图书在版编目(CIP)数据

控制数学问题的 MATLAB 求解/薛定宇，陈阳泉著．—北京：清华大学出版社，2007.11

（全国高等学校自动化专业系列教材）

ISBN 978-7-302-15297-2

Ⅰ．控…　Ⅱ．①薛…　②陈…　Ⅲ．数值计算－计算机辅助计算－软件包，MATLAB－高等学校－教材　Ⅳ．①TP391.75　②O241

中国版本图书馆 CIP 数据核字(2007)第 073864 号

责任编辑：王一玲
责任校对：时翠兰
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969
印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：175×245　**印　张**：31.25　**字　数**：624 千字
版　　次：2007 年 11 月第 1 版　　**印　　次**：2007 年 11 月第1 次印刷
印　　数：1～3000
定　　价：45.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：020838－01

出版说明

《全国高等学校自动化专业系列教材》

为适应我国对高等学校自动化专业人才培养的需要，配合各高校教学改革的进程，创建一套符合自动化专业培养目标和教学改革要求的新型自动化专业系列教材，"教育部高等学校自动化专业教学指导分委员会"（简称"教指委"）联合了"中国自动化学会教育工作委员会"、"中国电工技术学会高校工业自动化教育专业委员会"、"中国系统仿真学会教育工作委员会"和"中国机械工业教育协会电气工程及自动化学科委员会"四个委员会，以教学创新为指导思想，以教材带动教学改革为方针，设立专项资助基金，采用全国公开招标方式，组织编写出版了一套自动化专业系列教材——《全国高等学校自动化专业系列教材》。

本系列教材主要面向本科生，同时兼顾研究生；覆盖面包括专业基础课、专业核心课、专业选修课、实践环节课和专业综合训练课；重点突出自动化专业基础理论和前沿技术；以文字教材为主，适当包括多媒体教材；以主教材为主，适当包括习题集、实验指导书、教师参考书、多媒体课件、网络课程脚本等辅助教材；力求做到符合自动化专业培养目标、反映自动化专业教育改革方向、满足自动化专业教学需要；努力创造使之成为具有先进性、创新性、适用性和系统性的特色品牌教材。

本系列教材在"教指委"的领导下，从 2004 年起，通过招标机制，计划用 3～4 年时间出版 50 本左右教材，2006 年开始陆续出版问世。为满足多层面、多类型的教学需求，同类教材可能出版多种版本。

本系列教材的主要读者群是自动化专业及相关专业的大学生和研究生，以及相关领域和部门的科学工作者和工程技术人员。我们希望本系列教材既能为在校大学生和研究生的学习提供内容先进、论述系统和适于教学的教材或参考书，也能为广大科学工作者和工程技术人员的知识更新与继续学习提供适合的参考资料。感谢使用本系列教材的广大教师、学生和科技工作者的热情支持，并欢迎提出批评和意见。

《全国高等学校自动化专业系列教材》编审委员会

2005 年 10 月于北京

《全国高等学校自动化专业系列教材》编审委员会

序

FOREWORD

自动化学科有着光荣的历史和重要的地位，20 世纪 50 年代我国政府就十分重视自动化学科的发展和自动化专业人才的培养。五十多年来，自动化科学技术在众多领域发挥了重大作用，如航空、航天等，“两弹一星”的伟大工程就包含了许多自动化科学技术的成果。自动化科学技术也改变了我国工业整体的面貌，不论是石油化工、电力、钢铁，还是轻工、建材、医药等领域都要用到自动化手段，在国防工业中自动化的作用更是巨大的。现在，世界上有很多非常活跃的领域都离不开自动化技术，比如机器人、月球车等。另外，自动化学科对一些交叉学科的发展同样起到了积极的促进作用，例如网络控制、量子控制、流媒体控制、生物信息学、系统生物学等学科就是在系统论、控制论、信息论的影响下得到不断的发展。在整个世界已经进入信息时代的背景下，中国要完成工业化的任务还很重，或者说我们正处在后工业化的阶段。因此，国家提出走新型工业化的道路和“信息化带动工业化，工业化促进信息化”的科学发展观，这对自动化科学技术的发展是一个前所未有的战略机遇。

机遇难得，人才更难得。要发展自动化学科，人才是基础、是关键。高等学校是人才培养的基地，或者说人才培养是高等学校的根本。作为高等学校的领导和教师始终要把人才培养放在第一位，具体对自动化系或自动化学院的领导和教师来说，要时刻想着为国家关键行业和战线培养和输送优秀的自动化技术人才。

影响人才培养的因素很多，涉及教学改革的方方面面，包括如何拓宽专业口径、优化教学计划、增强教学柔性、强化通识教育、提高知识起点、降低专业重心、加强基础知识、强调专业实践等，其中构建融会贯通、紧密配合、有机联系的课程体系，编写有利于促进学生个性发展、培养学生创新能力的教材尤为重要。清华大学吴澄院士领导的《全国高等学校自动化专业系列教材》编审委员会，根据自动化学科对自动化技术人才素质与能力的需求，充分吸取国外自动化教材的优势与特点，在全国范围内，以招标方式，组织编写了这套自动化专业系列教材，这对推动高等学校自动化专业发展与人才培养具有重要的意义。这套系列教材的建设有新思路、新机制，适应了高等学校教学改革与发展的新形势，立足创建精品教材，重视实践性环节在人才培养中的作用，采用了竞争机制，以

激励和推动教材建设。在此，我谨向参与本系列教材规划、组织、编写的老师致以诚挚的感谢，并希望该系列教材在全国高等学校自动化专业人才培养中发挥应有的作用。

吴启迪教授

2005年10月于教育部

序

FOREWORD

《全国高等学校自动化专业系列教材》编审委员会在对国内外部分大学有关自动化专业的教材做深入调研的基础上，广泛听取了各方面的意见，以招标方式，组织编写了一套面向全国本科生（兼顾研究生）、体现自动化专业教材整体规划和课程体系、强调专业基础和理论联系实际的系列教材，自2006年起将陆续面世。全套系列教材共50多本，涵盖了自动化学科的主要知识领域，大部分教材都配置了包括电子教案、多媒体课件、习题辅导、课程实验指导书等立体化教材配件。此外，为强调落实“加强实践教育，培养创新人才”的教学改革思想，还特别规划了一组专业实验教程，包括《自动控制原理实验教程》、《运动控制实验教程》、《过程控制实验教程》、《检测技术实验教程》和《计算机控制系统实验教程》等。

自动化科学技术是一门应用性很强的学科，面对的是各种各样错综复杂的系统，控制对象可能是确定性的，也可能是随机性的；控制方法可能是常规控制，也可能需要优化控制。这样的学科专业人才应该具有什么样的知识结构，又应该如何通过专业教材来体现，这正是“系列教材编审委员会”规划系列教材时所面临的问题。为此，设立了《自动化专业课程体系结构研究》专项研究课题，成立了由清华大学萧德云教授负责，包括清华大学、上海交通大学、西安交通大学和东北大学等多所院校参与的联合研究小组，对自动化专业课程体系结构进行深入的研究，提出了按“控制理论与工程、控制系统与技术、系统理论与工程、信息处理与分析、计算机与网络、软件基础与工程、专业课程实验”等知识板块构建的课程体系结构。以此为基础，组织规划了一套涵盖几十门自动化专业基础课程和专业课程的系列教材。从基础理论到控制技术，从系统理论到工程实践，从计算机技术到信号处理，从设计分析到课程实验，涉及的知识单元多达数百个、知识点几千个，介入的学校50多所，参与的教授120多人，是一项庞大的系统工程。从编制招标要求、公布招标公告，到组织投标和评审，最后商定教材大纲，凝聚着全国百余名教授的心血，为的是编写出版一套具有一定规模、富有特色的、既考虑研究型大学又考虑应用型大学的自动化专业创新型系列教材。

然而，如何进一步构建完善的自动化专业教材体系结构？如何建设

基础知识与最新知识有机融合的教材？如何充分利用现代技术，适应现代大学生的接受习惯，改变教材单一形态，建设数字化、电子化、网络化等多元形态、开放性的“广义教材”？等等，这些都还有待我们进行更深入的研究。

本套系列教材的出版，对更新自动化专业的知识体系、改善教学条件、创造个性化的教学环境，一定会起到积极的作用。但是由于受各方面条件所限，本套教材从整体结构到每本书的知识组成都可能存在许多不当甚至谬误之处，还望使用本套教材的广大教师、学生及各界人士不吝批评指正。

吴澄 院士

2005年10月于清华大学

前言

PREFACE

控制学科中很多问题实际上就是应用数学问题，而控制学科的发展又催生了许多新的数学分支和方法，所以，利用一种强大的计算机数学语言来统一处理控制中的数学问题，对控制科学领域的学生与学者都有很积极的意义。这一方面可以从一个不同于以往的角度全面地认识控制理论，重新思考以往该学科中被忽视的实现问题，也有助于研究者借助计算机提供的强大功能探索新的知识，拓展新的研究方向。另一个方面，可以用简单的语句构造文献中的或自己提出的新方法，避免繁杂和极易出错的底层编程，用简洁的方式研究控制学科中的问题。比如说，在传统的控制学科中，连续系统的稳定性通常被建议采用 Routh 判据来判定，而离散系统应该采用更复杂的 Jury 表来判定，当然，这样的方法在控制理论的发展过程中起了里程碑式的作用，但随着计算机的发展和软件的进步，直接求取系统极点将比上述间接表格直观得多也容易得多，所以应该从观念上做出与时俱进的全面更新。另一个例子，纵观控制学科的文章会发现，诸多论著均声称提出的控制器效果优于工业上最常用的 PID 控制器，然而比较的 PID 控制器对象常用的是 20 世纪 40 年代提出的传统 Ziegler-Nichols 控制器，这样的比较当然有欠公允。所以从性能比较方面应该和效果最好的 PID 控制器相比较，才能得出有说服力的结论。但如何获得最好的 PID 控制器呢？如果不借助于强大的计算机工具是难以获得最好的 PID 控制器的，所以应该考虑与应用数学中的最优化问题求解结合起来，设定某些指标，然后用求解最优化的方法将最优控制器设计出来。要做到这一点，如果不掌握强大的计算机数学语言是不行的。

十年前，作者的著作《控制系统计算机辅助设计 —— MATLAB 语言与应用》由清华大学出版社出版。该书受到很多专家学者的关注，并被公认为国内关于 MATLAB 语言方面书籍中出版最早、影响最广的著作。该书被国内期刊文章和著作引用数千次，被数万篇硕士、博士论文引用，为我国高校师生和研究人员认识和掌握 MATLAB 语言，并用其解决控制领域各种各样的数学问题，受到各个层次读者的普遍欢迎。

本书继承了作者早期几部著作的优点，从使用者的角度出发，并结合作者十数年的实际编程经验和丰富的教学经验，系统地介绍各个数学分支中经典问题的MATLAB 求解，并将系统地介绍控制中问题的求解。其中一些内容取材于作者在清华大学出版社出版的其他两部著作《高等应用数学问题的 MATLAB 求解》和《控制系统计算机辅助设计 —— MATLAB 语言与应用》(第二版)，侧重数学问题在控制学科应用的特殊性，从结构上和叙述上做了大幅度的改进和整合，扩充了大量新的内容，将尤其适合于控制学科高年级本科生和研究生使用。

本书酝酿和写作过程中获得了控制界很多前辈、同行、朋友和学生的建议，作者特别感谢 Sussex 大学的 Derek P Atherton 教授、杨泰澄博士，东北大学任兴权教授、徐心和教授、刘建昌教授、井元伟教授、石海滨副教授，中科院系统科学研究院的韩京清研究员，清华大学郑大钟教授、胡东成教授、孙增圻教授、萧德云教授、王雄教授，哈尔滨工业大学马广富教授，北京交通大学朱衡君教授，新加坡国立大学葛树志教授，The MathWorks 公司的 Liang Jinsong 博士。东北大学潘峰、高道祥、赵春娜博士，博士生陈大力、鄂大志、张雪峰、翟春艳、曾静同学，硕士生熊琨等为本书部分内容的编写和软件设计做出了一定的工作。

本书承蒙东北大学张庆灵教授主审，并得到许多建设性建议。本书编写过程中一直得到本系列教材编委会副主任、清华大学萧德云教授和清华大学出版社前总编蔡鸿程老师的关注与帮助，本书的出版还得到了美国 The MathWorks 公司图书计划的支持，在此表示谢意，并特别感谢 Noami Fernandez 女士、Courtney Esposito 女士、Liz Salett 女士为作者提供的帮助。

由于作者水平所限，书中的缺点和错误在所难免，欢迎读者批评指教。

著 者

2007 年 2 月

教师反馈表

感谢您购买本书！清华大学出版社计算机与信息分社专心致力于为广大院校电子信息类及相关专业师生提供优质的教学用书及辅助教学资源。

我们十分重视对广大教师的服务，如果您确认将本书作为指定教材，请您务必填好以下表格并经系主任签字盖章后寄回我们的联系地址，我们将免费向您提供有关本书的其他教学资源。

您需要教辅的教材：			
您的姓名：			
院系：			
院/校：			
您所教的课程名称：			
学生人数/所在年级：	______人/ 1 2 3 4 硕士 博士		
学时/学期	______学时/______学期		
您目前采用的教材：	作者：______ 书名：______ 出版社：______		
您准备何时用此书授课：			
通信地址：			
邮政编码：		联系电话	
E-mail：			
您对本书的意见/建议：		系主任签字 盖章	

我们的联系地址：

清华大学出版社 学研大厦 A602，A604 室

邮编：100084

Tel：010-62770175-4409，3208

Fax：010-62770278

E-mail：liuli@tup.tsinghua.edu.cn；hanbh@tup.tsinghua.edu.cn

目录

CONTENTS

第 1 章 数学语言及其在控制中的应用概述

在控制科学学科中，控制理论的形成与发展是和数学的各个分支的进展是息息相关的。控制理论的发展也催生了若干新数学分支的出现与发展。在最早的反馈控制系统工程应用 —— 蒸汽机及 Watts 的飞锤控制器出现以后，科学家就开始研究这样系统的稳定性，从而将微分方程引入该系统的建模，并研究该系统的稳定性。在稳定性研究中出现了若干数学方法，如 Routh 判据及 Lyapunov 稳定性理论等。

由于微分方程求解比较麻烦，所以可以引入一种称为 Laplace 变换的积分变换，将微分方程变换成代数方程，由此引入了今天仍广泛使用的传递函数模型来描述线性微分方程模型。对传递函数模型还可以进行频域响应、时域响应与复域响应分析，得出系统的性质。

传递函数模型分析的是系统的外部性质，所以为了分析系统的内部性质还可以引入系统的状态方程表示，而该表示中需要大量的线性代数基础知识和求解方法。基于线性代数，研究者们提出了众多系统的设计方法，如基于 Riccati 方程求解的线性二次型最优控制方法、基于范数测度的 $\mathcal{H}_\infty$ 鲁棒控制器设计方法等。

如果想得到性能最好的控制器，则需要引入数学中的最优化技术，出现了最优控制的分支。

由前面的简单综述可以看出，控制科学中几乎所有的问题均可以归结成数学问题的求解，求解数学问题时手工推导当然是有用的，但并不是所有的问题都是能手工推导的，故需要由计算机来完成相应的任务。用计算机的方式也有两种，其一是用成型的数值分析算法、数值软件包与手工编程的方法相结合的求解方法，其二是采用国际上有影响的专门计算机语言来求解问题，这类语言包括 MATLAB、Mathematica、Maple 等，本书统一称之为**计算机数学语言**。顾名思义，用数值方法只能求解数值计算的问题，至于像公式推导等数学问题，例如求解 $x^3+ax+c=d$ 方程的解，在 a,c,d 不是给定数值时，数值分析的方式是没有用的，必须使用解析解的方法来求解。本书将数值解和解析解的求解统称为数学运算。

1.1 数学问题计算机求解概述

1.1.1 为什么要学习计算机数学语言

在系统介绍本书的内容之前，先介绍几个例子，读者可以思考其中提出的问题，从中体会掌握一个优秀的计算机数学语言的必要性。

例 1-1 控制系统的根轨迹是控制系统分析与设计的实用工具，假设某系统的开环传递函数模型为 $G(s)=\dfrac{0.6s^3+9.2s^2+11.9s+2.1}{s^4+2.8s^3+52.4s^2+68.7s+22.6}$，用传统控制课程中介绍的方法无法绘制根轨迹，因为根轨迹的起始点和终止点均未直接给出。利用一个实用的计算机数学语言，可以求出不同增益下的闭环特征根位置，从而绘制出系统的根轨迹。更直接的，用 MATLAB 控制系统工具箱提供的 `rlocus()` 函数可以直接绘制出系统的根轨迹，并根据根轨迹对系统进行直接分析。当然，若能求出系统的零极点，系统的稳定性分析就不在话下了。

例 1-2 试考虑含有非线性环节的控制系统模型，不借助计算机工具很难对系统进行仿真研究，因为这涉及到非线性微分方程的求解。利用 MATLAB 语言和其仿真环境 Simulink，这样的系统仿真研究轻而易举。

例 1-3 过程控制中经常遇到 PID 控制器的最优设计问题，给定控制系统模型，定义出需要最小化的目标函数，则可以通过寻优的方法得出最优控制器的参数。对这样的问题来说，掌握了最优化问题求解方法是很好解决的。此外，一般的最优化方法可能导致局部极小值，所以可以采用进化类方法，如遗传算法或粒子群方法，由现成的 MATLAB 工具可以直接求解，无需太多的底层算法和细节。

例 1-4 $\mathcal{H}_\infty$ 鲁棒控制计算的核心是 Riccati 方程的求解，这需要线性代数和方程求解方面的知识，有了 MATLAB 语言及其鲁棒控制工具箱，求解这样的问题就是几个命令直接调用的问题了。

从上面的例子可以看出，解决数学问题用手工推导的方法虽然有时可行，但对很多复杂问题是不现实或不可靠的，用传统数值分析方法甚至成型的软件包(如在国际上享有盛誉的 *Numerical Recipes*[1]) 得出的结果有时也是错误的，能够求解其部分内容而不是全部内容，所以需要学习计算机数学语言，以更好地解决以后学习和研究中遇到的问题。

1.1.2 数学问题的解析解与数值解

现代科学与工程的进展离不开数学。数学家们感兴趣的问题和其他科学家、工程技术人员所关注的问题是不同的。数学家往往对数学问题的解析解，或称闭式解 (closed-form solution) 和解的存在性严格证明感兴趣，而工程技术人员一般

对如何求出数学问题的解更关心。最直观的例子，回顾高等数学类课程经常有这样的定理：“……两点之间必定存在一点 ξ……”，而工程技术人员更侧重于如何找到 ξ，并找出满足某条件的全部 ξ，而不能只满足于其存在性。而获得这样解的最直接方法就是通过计算机数学语言。

解析解不存在的情况在数学上并不罕见，甚至可以说，这样的现象是常见的。例如，定积分 $\dfrac{2}{\sqrt{\pi}}\displaystyle\int_0^a \mathrm{e}^{-x^2}\mathrm{d}x$ 在上限为有穷时就没有解析解。数学家可以用新的函数 $\mathrm{erf}(a)$ 去定义这样的解，但解的值到底多大却不是一目了然的。所以，在这样的情况下，要想获得积分的值，就必须采用数值解技术。

再例如，圆周率 π 的值本身就没有解析解，中国古代的数学家、天文学家祖冲之早在公元 480 年就算定了该值在 3.1415926 和 3.1415927 之间。在一般科学与工程应用中，取这样的值就能保证较高的精度，而对于粗略估算来说，使用公元前 250 年 (?) 阿基米德的 3.1418 也未尝不可，而没有必要非去追求不存在的解析解不可。所以在这样的问题上，数值解法的优势就显示出来了。

数学问题的数值解法已经成功地应用于各个领域。例如，在力学领域，常用有限元法求解偏微分方程；在航空、航天与自动控制领域，经常用到数值线性代数与常微分方程的数值解法等解决实际问题；在工程与非工程系统的计算机仿真中，核心问题的求解也需要用到各种差分方程、常微分方程的数值解法；在高科技的数字信号处理领域，离散的快速 Fourier 变换 (FFT) 已经成为其不可或缺的工具。在科学工程研究中能掌握一个或多个实用的计算工具，无疑会为研究者提供解决实际问题的强有力手段。

1.1.3 数学运算问题软件包发展概述

数字计算机的出现给数值计算技术的研究注入了新的活力。在数值计算技术的早期发展中，出现了一些著名的数学软件包，如美国的基于特征值的软件包 EISPACK [2,3]和线性代数软件包 LINPACK [4]， 英国牛津数值算法研究组 (Numerical Algorithm Group，NAG) 开发的 NAG 软件包 [5]及享有盛誉的著作 *Numerical Recipes*[1]中给出的程序集等，这些都是在国际上广泛流行的、有着较高声望的软件包。

美国的 EISPACK 和 LINPACK 都是基于矩阵特征值和奇异值解决线性代数问题的专用软件包。限于当时的计算机发展状况，这些软件包大都是由 Fortran 语言编写的源程序组成的。

例如，若想求出 N 阶实矩阵 $\boldsymbol{A}$ 的全部特征值 (用 $\boldsymbol{W}_{\mathrm{R}}$，$\boldsymbol{W}_{\mathrm{I}}$ 数组分别表示其实虚部) 和对应的特征向量矩阵 $\boldsymbol{Z}$，则 EISPACK 软件包给出的子程序建议调用路径为

```
CALL BALANC(NM,N,A,IS1,IS2,FV1)
```

```
CALL ELMHES(NM,N,IS1,IS2,A,IV1)
CALL ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
CALL HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
IF (IERR.EQ.0) GOTO 99999
CALL BALBAK(NM,N,IS1,IS2,FV1,N,Z)
```

由上面的叙述可以看出，要求取矩阵的特征值和特征向量，首先要给一些数组和变量依据 EISPACK 的格式作出定义和赋值，并编写出主程序，再经过编译和连接过程，形成可执行文件，最后才能得出所需的结果。

英国的 NAG 软件包和美国学者的 *Numerical Recipes* 工具包则包括了各种各样数学问题的数值解法，二者中 NAG 的功能尤其强大。NAG 的子程序都是以字母加数字编号的形式命名的，非专业人员很难找到适合自己问题的子程序，更不用说能保证以正确的格式去调用这些子程序了。这些程序包使用起来极其复杂，谁也不能保证不发生错误，NAG 数百页的使用手册就有十几本之多!

Numerical Recipes 一书中给出的一系列算法语言源程序也是一个在国际上广泛应用的软件包。该书中的子程序有 C, FORTRAN 和 Pascal 等版本，适合于科学研究者和工程技术人员直接应用。该书的程序包由 200 多个高效、实用的子程序构成，这些子程序一般有较好的数值特性，比较可靠，为科学研究者所信赖。

具有 Fortran 和 C 等高级计算机语言知识的读者可能已经注意到，如果用它们去进行程序设计，尤其当涉及矩阵运算或画图时，则编程会很麻烦。例如，若想求解一个线性代数方程，用户得首先去编写一个主程序，然后编写一个子程序去读入各个矩阵的元素，之后再编写一个子程序，求解相应的方程 (如使用 Gauss 消去法)，最后输出计算结果。如果选择的计算子程序不是很可靠，则所得的计算结果往往可能会出现问题。如果没有标准的子程序可以调用，则用户往往要将自己编好的子程序逐条地输入计算机，然后进行调试，最后进行计算。这样一个简单的问题往往需要用户编写 100 条左右的源程序，输入与调试程序也是很费事的，并无法保证所输入的程序完全可靠。求解线性方程组这样一个简单的功能需要 100 条源程序，其他复杂的功能往往要求有更多条语句，如采用双步 QR 法求取矩阵特征值的子程序则需要 500 多条源程序，其中任何一条语句有毛病，甚至调用不当 (如数组维数不匹配) 都可能导致错误结果的出现。

尽管如此，数学软件包仍在继续发展，其发展方向是采用国际上最先进的数值算法，提供更高效、更稳定、更快速、更可靠的数学软件包。例如，在线性代数计算领域，全新的 LaPACK 已经成为当前最有影响的软件包，但它们的目的似乎已经不再是为一般用户提供解决问题的方法，而是为数学软件提供底层的支持。新版的 MATLAB 语言以及自由软件 Scilab 等著名的计算机数学语言都已经放弃了一直使用的 LINPACK 和 EISPACK 软件包，而采用 LaPACK 为其底层支持软件包。

一些数学的专门分支也出现了相关的数学程序库，支持 Fortran, C++ 等语言直接调用与编程。在互联网上同样有大量的 MATLAB 语言和其他计算机数学语言的数学工具箱，所以遇到典型问题的数学求解时，可以直接利用相关的工具箱来求解，因为其中大部分工具箱毕竟还是在相应领域有影响的专家编写的，得出的结果比外行自己查阅书籍、论文编写的可信度要高得多。

1.1.4 代表性计算机数学语言

目前在国际上有 3 种计算机数学语言最有影响：The MathWorks 公司的 MATLAB 语言、Wolfram Research 公司的 Mathematica 语言和 Waterloo Maple 公司的 Maple 语言。这 3 种语言各有特色，其中 MATLAB 长于数值运算，其程序结构类似于其他计算机语言，因而编程很方便。Mathematica 和 Maple 有强大的解析运算和数学公式推导、定理证明的功能，相应的数值计算能力比 MATLAB 要弱，这两个语言更适合于纯数学领域的计算机求解。

和 Mathematica 及 Maple 相比，MATLAB 语言的数值运算功能是很出色的。除此之外，更有一个其他两种语言不可替代的优势，就是 MATLAB 语言对各种各样领域均有领域专家编写的工具箱，可以高效、可靠地解决各种各样的问题。MATLAB 的符号运算工具箱利用 Maple 作为其符号运算引擎，能直接求解常用的符号运算问题。另外，MATLAB 提供了对 Maple 全部函数的接口，无需安装 Maple 就可以调用 Maple 所有的数学函数，这大大地增强了 MATLAB 的符号运算功能，在这方面的功能也不逊色于 Mathematica 和 Maple。MATLAB 的仿真工具 Simulink 可以利用框图方法对各种工程系统进行方便的仿真研究，MATLAB 语言覆盖的学科之广、功能之全是其他两种语言无法比拟的。故本书采用 MATLAB 语言为主要计算机数学语言，系统介绍其在数学问题求解中的应用。掌握了该语言将提高读者求解数学问题的能力，提高数学水平，拓广知识面，使得原来看起来无从下手的高深应用数学问题的实际求解变得轻而易举。

1.2 控制领域对数学问题的依赖

1.2.1 控制领域计算机软件包发展概述

国际上控制系统计算机辅助设计 (computer aided control systems design，CACSD) 软件的发展大致分为几个阶段：软件包阶段、交互式语言阶段及当前的面向对象的程序环境阶段[6]。

在早期的工作中，CACSD 主要集中在软件包的编写上，1973 年美国学者 Melsa 教授和 Jones 博士出版了一本专著[7]，书中给出了许多当时流行的控制系统计算机辅助分析与设计的源程序，包括求取系统的根轨迹、频域响应、时间响

应，以及各种控制系统设计的子程序如 Luenberger 观测器、Kalman 滤波等。瑞典 Lund 工学院 Karl Åström 教授主持开发的一套交互式控制系统计算机辅助设计软件 INTRAC (IDPAC, MODPAC, SYNPAC, POLPAC 等，以及仿真语言 SIMNON) [8]，其中的 SIMNON 仿真语言要求用户依照它所提供的语句编写一个描述系统的程序，然后才可以对控制系统进行仿真。日本的古田胜久 (Katsuhisa Furuta) 教授主持开发的 DPACS-F 软件 [9]，在处理多变量系统的分析和设计上还是很有特色的。在国际上流行的仿真语言 ACSL, CSMP, TSIM, ESL 等也同样要求用户编写模型程序，并提供了大量的模型模块。在这一阶段还出现了很多的专用程序，如英国剑桥大学推出的线性系统分析与设计软件 CLADP (Cambridge linear analysis and design programs) [10,11]与美国国家宇航局 (NASA) Langley 研究中心的 Armstrong 开发的线性二次型最优控制器设计的 ORACLS (optimal regulator algorithms for the control of linear systems) [12]等。

在早期的工作中，CACSD 主要集中在软件包的编写上，如前面提及的 Melsa 和 Jones 的著作。从数值算法的角度上也出现了一些著名的软件包，如美国的基于特征值的软件包 EISPACK [2,3]和线性代数软件包 LINPACK [4]，英国牛津数值算法研究组 (Numerical Algorithm Group) 开发的 NAG 软件包 [5]及文献 [1] 中给出的声誉颇高的数值算法程序集等，在 CACSD 领域的经典软件包作品有英国 Kingston Polytechnic 控制系统研究组开发的 SLICE (subroutine library in control engineering) 软件包 [13]，前面提及的 DPACS-F, ORACLS 等。这些软件包大都是由 FORTRAN 语言编写的源程序组成的，给使用者提供了较好的接口，但和 MATLAB 相比，调用方法和使用明显显得麻烦、不便。此外，以前 FORTRAN 语言绘图并不是轻而易举的事情，这就需要再调用相应的软件包来做进一步处理，在绘图方面比较实用和流行的软件包是 GINO-F [14]，但这种软件包只给出绘图的基本子程序，所以要绘制较满意的图形需要用户自己去用这些低级命令去编写出合适的绘图子程序来。

20 世纪 70 年代末期和 80 年代初期出现了很多实用的具有良好人机交互功能的软件，MATLAB 就是其中的一个成功的范例，此外前面提及的 INTRAC 和 CTRL-C 等也是优秀的人机交互式软件。

正因为存在多种多样的 CACSD 软件，而它们之间又各有所长，所以在 CACSD 技术的发展过程中曾有过几次将若干常用软件集成在一起的尝试，例如 1984 年前后美国学者 Spang III 教授曾将当时流行的 SIMNON, CLAPD, IDPAC 及他自己研制的 SSDP (state space design program) 集成在一起，形成了一个强大的软件 [15]，各个组成软件之间是靠读写文件的方式来传递数据的，这多少可以解决前面提及的程序之间不能传递数据的弊病。1986 年前后由英国科学与工程委员会 (SERC) 资助，Harold Rosenbrock 教授和 Neil Munro 教授主持的、英国多所大学和研究机构参与的 ECSTASY (environment for control

system theory and synthesis) 软件环境的开发项目[16]，在该软件中试图将流行的新一代软件如 MATLAB, ACSL, TSIM 甚至当时刚出现的 Mathematica[17] 等集成到一个框架之下，该软件还可以同时自动采用 LaTeX[18] 和 FrameMaker 等来输出专业的排版结果，并取得了一些成效。各个软件之间的数据传递是通过数据库来实现的，ECSTASY 定义了方便实用的 CACSD 新命令，比当时的 MATLAB 更为简洁。ECSTASY 这样的软件是一个有益的尝试，但该软件当时只可以在 Sun 工作站上运行，并没有考虑 PC 的兼容性。

依作者之见，这些集成出来的软件并不是很成功的，因为它们并没有达到预期的效果。事实上，从那以后每个软件的功能都有了明显的改善，MATLAB 语言有了自己的仿真功能，Simulink 从某种意义上来讲其功能和接口更优于 ACSL, MATLAB 和 Mathematica 之间也有了较好的接口，它们的优势可以得到充分的互补。

我国较有影响的控制系统仿真与计算机辅助设计成果是中科院系统科学研究所韩京清研究员等主持的国家自然科学基金重大项目开发的 CADCSC 软件[19]和清华大学孙增圻、袁曾任教授的著作和程序[20]，以及北京化工学院吴重光、沈成林教授的著作和程序[21]等。

1.2.2 MATLAB 和其他语言在控制领域应用的比较

在控制工程领域，Mathematica 有自己的 Control Systems Professional 程序集，可以较好地解决一些控制中的问题[22]。另外，一些有影响的学者也开发了一个自由软件控制工具 Slicot，可以较好地解决一般控制问题与鲁棒控制问题[23]。

随着控制理论的发展，现在控制科学与工程学科所覆盖的内容已经远不是传统意义下的控制系统仿真与设计等内容了，新的控制研究分支也不断出现，用前面提及的 Control Professional 和 Slicot 已远远不能求解这些控制的问题了。另一方面，The MathWorks 不断推出新的控制类工具箱，而隶属于 The MathWorks 的用户文件交互网站[1] 也不断发布新的用户自行开发的工具箱，其中不乏优秀、实用的程序。MATLAB 在控制领域发挥着和其他软件相比越来越重要的作用。

1.2.3 控制中的数学问题

本书将从三个层面上介绍控制系统数学问题的求解方法。

① **系统介绍各个数学分支现有问题的计算机求解**：本书将按照不同的学科系统介绍微积分与积分变换、线性代数、微分方程、最优化等数学问题的求解方法，并演示这些分支中数学问题求解的实例。另外，将对不同的控制

[1] 地址：`http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do`。

理论研究领域，如差分方程与自适应控制、智能控制与智能计算、鲁棒控制和分数阶控制等专门领域，系统介绍数学问题的求解方法和基于工具箱的直接求解方法，为读者更好理解这些领域问题和研究奠定较好的基础。

② **用较底层的编程实现典型算法**：本书除了介绍很多数学问题的求解语句之外，对典型的问题，如 Lyapunov 方程的解析求解、自校正控制器设计、$\mathcal{H}_\infty$ 范数的求解等，底层的简洁编程语句有助于读者更好地理解所介绍的算法。

③ **利用计算机开拓新的领域**：充分利用 MATLAB 强大的数值与解析运算功能，编写出以往不存在的新的函数，如矩阵任意函数求值、最优控制器设计程序开发、分数阶控制系统分析框架建立等。掌握了这些方法，读者可以更好地在自己的学习与研究中使用 MATLAB 语言及工具，解决自己的问题，创造性地开拓新的研究。

1.2.4 依赖计算机数学语言的控制研究新观念

传统的反馈控制理论已经是一门很成型的课程了，但从目前的发展看，已经不能反映现在学科发展的需要。例如，在控制系统分析领域，仍然有传递函数与状态方程、连续与离散、单变量与多变量等明显的分界，缺乏一个统一的系统分析框架。MATLAB 的控制系统工具箱提供了统一的控制系统分析框架，用统一的求解语句对系统进行有效分析。

传统反馈控制理论中很多内容和方法已经是不合时宜的了，比如有一个系统模型，若想分析其稳定性，传统教学方法会让研究者首先想到建立系统的 Routh 表。Routh 表的出现当然是控制理论发展的一个重要的里程碑，不过 Routh 表的出现是受当时技术和工具发展现状制约的，随着 MATLAB 这类计算机工具的出现，求出控制系统极点远比构造 Routh 表容易得多也直观得多。另外，在离散系统和多变量系统的稳定性分析中，极点求解方法远远优于 Routh 表、Jury 表[24]这类间接判定方法。

线性系统的频域分析在反馈控制系统的分析与设计中无疑起着巨大的作用，然而传统的 Nyquist 曲线有一个很致命的缺陷，在 Nyquist 图上缺失增益与频率的对应关系，所以会对控制系统的设计有一定的制约。用 MATLAB 提供的频域响应曲线可以弥补这方面的缺陷，更好地设计所需的控制系统。

传统控制理论中，最优控制器设计更偏重于二次型指标的控制与调节，因为这样的设计方法可能导致易于求解的数学问题，如代数 Riccati 方程的求解。然而这样的最优模型建立方法显然严重偏离了控制的要求，因为追求控制问题的可解性，牺牲了控制问题的物理意义。在跟踪控制中，跟踪的偏差是最重要的控制指标之一，而二次型控制很难将这样的要求纳入定型的目标函数框架，所以得出的解不一定有实际意义。

非线性系统仿真与设计是传统反馈控制理论中的一个难点，对日益需要广泛探讨的非线性系统来说，传统的方法只提供了相空间方法与描述函数方法这类近似的分析方法[25]。MATLAB/Simulink 的出现为非线性系统的仿真研究提供了可靠的手段，理论上可以对任意复杂度的系统进行分析。

1.3 MATLAB 语言预备知识

1.3.1 MATLAB 语言简介

1984 年正式推出的 MATLAB 语言为数学问题的计算机求解，特别是控制系统的仿真和 CAD 发展起到了巨大的推动作用。1980 年前后，时任美国 New Mexico 大学计算机科学系主任的 Cleve Moler 教授认为用当时最先进的 EISPACK 和 LINPACK 软件包求解线性代数问题过程过于烦琐，所以构思一个名为 MATLAB (MATrix LABoratory，即矩阵实验室) 的交互式计算机语言[26]。该语言 1980 年出现了免费版本，1984 年 The MathWorks 公司成立，并推出了 1.0 版。该语言的出现正赶上控制界基于状态空间的控制理论蓬勃发展的阶段，所以很快就引起了控制界学者的关注，出现了用 MATLAB 语言编写的控制系统工具箱，在控制界产生了巨大的影响，成为控制界的标准计算机语言。后来由于控制界及相关领域提出的各种各样要求，MATLAB 语言得到了持续发展，使得其功能越来越强大。可以说，MATLAB 语言是由计算数学专家首创的，但却是由控制界学者“捧红”的新型计算机语言。目前大部分工具箱都是面向控制和相关学科的，但随着 MATLAB 语言的不断发展，目前在其他领域已经开始了越来越广泛的应用，逐渐成为好多领域的首选计算机数学语言。

1.3.2 基本数据类型与基本语句结构

1. 基本数据类型

强大方便的数值运算功能是 MATLAB 语言的最显著特色。为保证较高的计算精度，MATLAB 语言中最常用的数值量为双精度浮点数，占 8 个字节 (64 位)，遵从 IEEE 记数法，有 11 个指数位、53 位尾数及一个符号位，值域的近似范围为 -1.7×10^{308} 至 1.7×10^{308}，其 MATLAB 表示为 `double()`。考虑到一些特殊的应用，比如图像处理，MATLAB 语言还引入了无符号的 8 位整型数据类型，其 MATLAB 表示为 `uint8()`，其值域为 $0\sim255$，这样可以大大地节省 MATLAB 的存储空间，提高处理速度。此外，在 MATLAB 中还可以使用其他的数据类型，如 `int8()`，`int16()`，`int32()`，`uint16()`，`uint32()` 等，每一个类型后面的数字表示其位数，其含义不难理解。

MATLAB 提供的符号运算工具箱还支持符号变量的使用，符号型数据类型

在公式推导与数学问题解析解中有非常重要的意义。可以由 `syms` a b c 来申明符号变量 a,b,c。

除了常用的数值型数据外，MATLAB 还支持字符串、多维数组、结构图、类与对象等各种数据类型。这些数据类型在后面的叙述中将详细介绍。

2. 变量与常量

MATLAB 语言变量名应该由一个字母引导，后面可以跟字母、数字、下划线等。例如，`MYvar12`, `MY_Var12` 和 `MyVar12_` 均为有效的变量名，而 `12MyVar` 和 `_MyVar12` 为无效的变量名。在 MATLAB 中变量名是区分大小写的，也就是说，`Abc` 和 `ABc` 两个变量名表达的是不同的变量，在使用 MATLAB 语言编程时一定要注意。

MATLAB 允许各种常数，如 `eps` 为机器的浮点运算误差限。PC 上 `eps` 的默认值为 2.2204×10^{-16}。`Inf` 为无穷大量 $+\infty$ 的 MATLAB 表示。`NaN` 为不定式 (not a number) 表示。`pi` 为圆周率 π 的双精度浮点表示。在实际编程时，这些常量可以直接使用。

3. 矩阵输入语句

矩阵的输入在 MATLAB 下是非常简单、直观的。下面通过例子演示实数、复数矩阵的输入方法。

例 1-5 试输入下面两个矩阵

$$\boldsymbol{A}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}4+2\mathrm{j}&2+4\mathrm{j}&2+2\mathrm{j}\\1+3\mathrm{j}&4+3\mathrm{j}&4\mathrm{j}\\3+3\mathrm{j}&3&4+4\mathrm{j}\end{bmatrix}$$

求解 由下面的 MATLAB 语句可以容易地输入这两个矩阵

```
>> A=[1,2,3; 4,5,6; 7,8,0];
   B=[4+2i,2+4i,2+2i; 1+3i,4+3i,4i; 3+3i,3,4+4i];
```

其中的 >> 为 MATLAB 的提示符，由机器自动给出，在提示符下可以输入各种各样的 MATLAB 命令。矩阵的内容由方括号括起来的部分表示，在方括号中的分号表示矩阵的换行，逗号或空格表示同一行矩阵元素间的分隔。给出了上面的命令，就可以在 MATLAB 的工作空间中建立 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 两个变量了。如果不想显示中间结果，则应该在语句末尾加一个分号。

注意，虽然 MATLAB 定义了 i 和 j 均表示 $\sqrt{-1}$，但在程序设计中这两个变量可能被改写，所以最好不采用 `3*i` 这样的语句，而 i 本身可以写成 `1i`。

一个数值矩阵用 `sym()` 函数进行处理，则可以将其变换成符号型矩阵。对进一步的矩阵运算来说，符号型矩阵将采用解析方法，而数值型矩阵采用数值方法。

4. 函数调用语句

MATLAB 函数形式是 MATLAB 程序设计的主流形式，其基本格式为

```
[返回变量列表]=func_name(输入变量列表)
```

其中，语句左侧为函数返回的变量，若同时返回多个变量，则应该用方括号括起来。输入变量在右侧给出。例如，函数 bode() 可以由 $[m,p]$=bode(G,ω) 语句格式调用，该语句表示由系统模型 G 和给定频率向量 ω 可以计算出系统的频率响应幅值 m 和相位 p。MATLAB 还允许一个函数有多种调用格式，例如 bode() 函数可以由几种格式调用 bode(G,ω)， bode(G)， bode(G_1,G_2,G_3) 。

函数可以按照不同的数据类型进行调用，例如 diff() 函数本身有多种定义，可以在很多目录查找到 diff.m 文件。其中，在 @sym 目录的文件将对符号变量进行运算，可以对给定的函数进行微分运算，而 MATLAB 本身的 diff() 函数对已知数值向量进行差分运算。

正是因为支持了函数的不同调用格式，也支持了不同对象模型的重载函数定义，才使得 MATLAB 语言以适合于控制系统分析的统一框架建立。例如，可以用一个变量 G 来表示不同的对象类型，例如可以用传递函数、状态方程等，可以描述连续、离散模型，可以是单变量或多变量的，可以是带有时间延迟或不带有时间延迟的，总之可以由某种方式描述出 G，这时，系统的 Bode 图可以由统一的命令 bode(G) 来绘制。

1.3.3 流程控制结构简介

MATLAB 支持的主要流程控制语句是循环结构、转移结构、开关结构和试探结构等。这些流程的基本结构简述如下：

① **循环结构 1** for i=v，循环体语句段，end

② **循环结构 2** while (表达式)，循环体语句段，end

③ **转移结构**

```
if (表达式 1)，语句段 1，
elseif (表达式 2)，语句段 2，
   ⋮
else，语句段，end
```

④ **开关结构**

```
switch 开关表达式
case 表达式 1， 语句段 1
case {表达式 2,表达式 3,⋯， 表达式 m}，语句段 2
   ⋮
otherwise， 语句段 n， end
```

⑤ **试探结构** try， 语句段 1，catch， 语句段 2，end

1.3.4 MATLAB 语言和 C 语言的对比实例

下面将给出两个实际编程例子演示，对 C 这类编程语言来说，解决同样的数学运算问题与控制问题需要大量的底层编程，而这些底层编程存在各种隐患。如果有个别细节考虑不周，则得出的结果可能是错误的。所以应该采用更可靠、更简洁的专门计算机数学语言来进行科学研究，因为这样可以将研究者从烦琐的底层编程中解放出来，更好地把握要求解的问题，避免“只见树木、不见森林”的现象，这无疑是受到更多研究者认可的方式。

例 1-6 已知 Fibonacci 数列的前两个元素为 $a_1 = a_2 = 1$，随后的元素可以由 $a_k = a_{k-1} + a_{k-2}, k = 3, 4, \cdots$ 递推地计算出来。试用计算机列出该数列的前 100 项。

求解 C 语言在编写程序之前需要首先给变量选择数据类型，如此问题需要的是整数，所以很自然地选择 int 或 long 来表示数列的元素，若选择数据类型为 int，则可以编写出如下 C 程序

```
main()
{  int a1, a2, a3, i;
   a1=1; a2=1; printf("%d  %d  ",a1,a2);
   for (i=3; i<=100; i++)
   {  a3=a1+a2; printf("%d  ",a3); a1=a2; a2=a3;
}}
```

只用了上面几条语句，问题就看似轻易地被解决了。然而该程序是错误的！运行该程序会发现，该数列打印到第 24 项突然会出现负数，而再显示下几项会发现时正时负。显然，上面的程序出了问题。问题出在 int 整型变量的选择上，因为该数据类型即使采用 long 整型数据定义，也只能保留 31 位二进制数值，即保留 9 位十进制有效数字，超过这个数仍然返回负值。可见，采用 C 语言，如果某些细节考虑不到，则可能得出完全错误的结论。故可以说 C 这类语言得出的结果有时不大令人信服。用 MATLAB 语言则不必考虑这些烦琐的问题。

```
>> a=[1 1]; for i=3:100, a(i)=a(i-1)+a(i-2); end; a
```

另外，由于 long 整型数据只能保持 9 位有效数字，而 double 型只能保留 15 位有效数字，如果得出的结果超出此范围，则精度将存在局限性。采用 MATLAB 的符号运算则可以避免这类问题，只需将第一个语句修改成 a=sym([1,1]) 就可以得出 a_{100} 的值为 354224848179261915075，这个结果是采用任何数值运算无法得出的。

例 1-7 试编写出两个矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 相乘的 C 语言通用程序。

求解 如果 $\boldsymbol{A}$ 为 $n \times p$ 矩阵，$\boldsymbol{B}$ 为 $p \times m$ 矩阵，则由线性代数理论，可以得出 $\boldsymbol{C}$ 矩阵，其元素为 $c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$，$i = 1, \cdots, n$, $j = 1, \cdots, m$。分析上面的算法，容易编写出 C 语言程序，其核心部分由三重循环结构计算出

```
for (i=0: i<n; i++){ for (j=0; j<m; j++){
    c[i][j]=0;
    for (k=0; k<p; k++) c[i][j]+=a[i][k]*b[k][j];
}}
```

看起来这样一个通用程序通过这几条语句就解决了。事实不然，这个程序有个致命的漏洞，就是没考虑两个矩阵是不是可乘。通常我们知道，两个矩阵可乘，则 $\boldsymbol{A}$ 矩阵的列数应该等于 $\boldsymbol{B}$ 的行数，所以应该加一个判定语句

if $\boldsymbol{A}$ 的列数不等于 $\boldsymbol{B}$ 的行数，给出错误信息

其实这样的判定也有漏洞，因为若 $\boldsymbol{A}$ 或 $\boldsymbol{B}$ 为标量，则 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 无条件可乘，而增加上面的 if 语句反而会给出错误信息。这样在原来的基础上还应该增加判定 $\boldsymbol{A}$ 或 $\boldsymbol{B}$ 是否为标量的语句。

其实即使考虑了上面所有的内容，程序还不是通用的程序，因为并未考虑矩阵为复数矩阵的情况。这也需要特殊的语句处理。

从这个例子可见，用 C 这类语言处理某类标准问题时需要特别细心，否则难免会有漏洞，致使程序出现错误，或其通用性受到限制。在 MATLAB 语言中则没有必要考虑这样的琐碎问题，因为 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的积由 $\boldsymbol{A}*\boldsymbol{B}$ 直接求取，若可乘则得出正确结果，如不可乘则给出出现问题的原因。

当然，在实时性与实时控制等领域，C 语言也有它的优势。当然 MATLAB 的代码也可以自动翻译成 C 语言程序，这不是本书叙述的范围。

1.3.5 图形绘制

假设用户已经获得了一些实验数据。例如，已知各个时刻 $t=t_1,t_2,\cdots,t_n$ 和在这些时刻的函数值 $y=y_1,y_2,\cdots,y_n$，则可以将这些数据输入到 MATLAB 环境中，构成向量 $\boldsymbol{t}=[t_1,t_2,\cdots,t_n]$ 和 $\boldsymbol{y}=[y_1,y_2,\cdots,y_n]$，如果用户想用图形的方式表示二者之间的关系，则给出 plot($\boldsymbol{t}$,$\boldsymbol{y}$) 命令即可绘制二维图形。plot() 函数的最一般调用格式为

plot($\boldsymbol{t}_1,\boldsymbol{y}_1$,选项 1, $\boldsymbol{t}_2,\boldsymbol{y}_2$,选项 2, $\cdots$, $\boldsymbol{t}_m,\boldsymbol{y}_m$,选项 m)

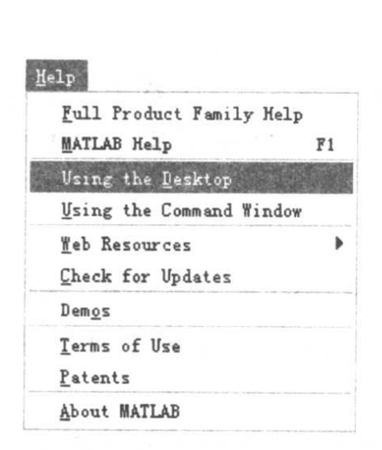
二维图形还可以有其他表示形式，如 stairs() 函数可以绘制出阶梯折线，semilogx() 可以绘制出横坐标为对数坐标的曲线，这些曲线在控制系统分析中有一定的实际意义。

MATLAB 语言还可以容易地绘制三维图形，例如 surf() 函数可以绘制出三维表面图，mesh() 可以绘制三维网格图。

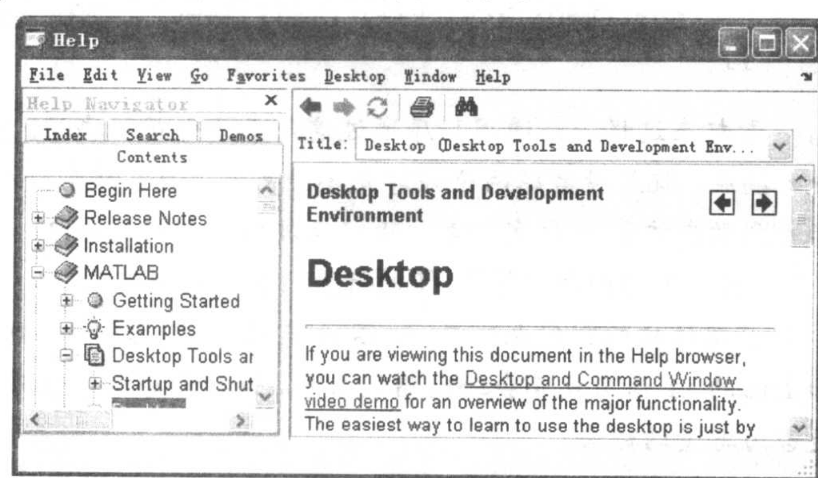
1.3.6 联机帮助信息

联机帮助信息可以由 MATLAB 命令窗口的 Help 菜单获得，该菜单将打开

如图 1-1 (a) 所示的菜单项。选择其中的 Using the Desktop 菜单项将打开联机帮助信息窗口，如图 1-1 (b) 所示。帮助信息可以由该窗口得出。



(a) Help 菜单



(b) 联机帮助信息窗口

图 1-1　联机帮助信息查询

还可以在 MATLAB 命令窗口下键入 `help` 命令或 `doc` 直接显示帮助信息。还可以使用 `lookfor` 命令查询某关键词，获得帮助信息。

1.4　本书框架设计及内容安排

本书首先给出数学问题计算机求解及控制数学问题求解的概述，着重介绍 MATLAB 语言的发展简况和编程必备基础知识，关于 MATLAB 编程的进一步知识可以在后续章节中介绍，也可以参考其他文献，如 [27,28]。学好 MATLAB 语言，可以将 30 字的学习准则作为座右铭，即“带着问题学，活学活用，学用结合，急用先学，立竿见影，在用字上狠下功夫”。

第 2 章首先给出微积分问题的计算机求解方法，并介绍给定数学函数的 Taylor 级数和 Fourier 级数拟合及拟合效果研究，然后介绍 Laplace 变换、Fourier 变换及其反变换，引入控制系统模型的表示方法，还将介绍控制系统模型的互联与化简方法。

第 3 章首先介绍特殊矩阵的简洁输入方法，然后介绍矩阵的性质分析、矩阵分解、矩阵方程求解和矩阵函数求解方法。并将介绍矩阵分析与变换在控制系统中的应用，各种矩阵方程如 Lyapunov 方程和 Riccati 方程的求解方法。该章还给出了矩阵函数求解的一般算法及函数，可以求解任意复杂的矩阵函数。

第 4 章介绍微分方程的计算机求解方法。首先给出线性微分方程的解析解算法，然后介绍微分方程的数值求解算法、MATLAB 实现、方程变换方法与解检验的方法，还介绍了各种特殊微分方程的求解方法，如刚性方程求解、隐式微分方程求解、微分代数方程求解等内容，介绍微分方程边值问题的求解方法，并介绍

基于 Simulink 的各种控制系统微分方程求解方法。

第 5 章介绍代数方程求解与最优化问题求解方法，在代数方程求解中将介绍多项式方程的准解析解算法和一般代数方程的数值解方法。在最优化问题求解方面首先介绍无约束最优化问题的求解方法，然后介绍线性规划问题的求解方法，二次型规划的求解方法和一般非线性规划问题的求解方法。本章还探讨了整数规划与混合整数规划问题的求解方法，研究各种与最优化问题相关的控制问题，如系统线性化问题、模型最优降阶问题等，并给出了一个通用的最优控制器设计程序，还探讨了 Minimax 方法在控制器设计中的应用。

第 6 章侧重介绍离散系统的分析与设计方法，首先给出离散系统的定义与分类方法，介绍非线性系统的仿真方法以及离散系统的辨识方法。还将介绍自校正控制器的设计与仿真、预测控制、广义预测控制等的设计与仿真方法。

第 7 章介绍智能计算和智能控制系统的方面与仿真问题，首先介绍模糊集合、模糊决策的基本概念，给出模糊逻辑控制器的设计方法与仿真方法，然后介绍神经网络原理及其在数据拟合中的应用，并给出基于神经网络的预测控制和模型参考控制的仿真。本章还介绍遗传算法和粒子群算法在最优化求解问题以及在最优控制器设计中的应用。最后介绍迭代学习控制的仿真方法。

第 8 章介绍鲁棒控制问题的数学基础与求解方法，首先介绍不确定系统的描述方法，然后介绍 $\mathcal{H}_\infty$ 和 $\mathcal{H}_2$ 控制的设计方法以及线性矩阵不等式的设计方法。该章还介绍定量反馈理论在鲁棒控制器设计中的应用，并给出基于状态反馈的多变量系统解耦设计方法。

第 9 章介绍分数阶微积分及分数阶系统的研究方法，首先给出各种分数阶微积分的定义与计算方法，然后引入分数阶传递函数类的构造，并以重载的方式介绍了分数阶系统的互联、时域与频域分析方法。本章还将介绍 Oustaloup 滤波器及其改进形式的分数阶微分算子的拟合方法以及基于框图的分数阶非线性微分方程求解方法。最后研究分数阶系统的次最优降阶方法与控制器设计领域的研究方法。

1.5 习题与思考题

1 The MathWorks 网站 (`http://www.mathworks.com`) 上提供了 MATLAB 及所有工具箱手册的电子版，如果需要，可以将感兴趣的工具箱手册下载阅读。由于工具箱规模过于庞大，不可能在一本书中全盘介绍，所以本书的作用只是作为读者学习和使用 MATLAB 语言的入门材料。

2 MATLAB Central 网站

`http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do`

包含大量由 MATLAB/Simulink 用户编写的实用程序和模型，其中不乏高水平的

作品。许多 MATLAB 本身没有直接提供的功能均可以从这里找到解决方法，试浏览该网站提供的各类程序。

3 MATLAB 语言是控制系统研究的首选语言，本书以该语言为主线介绍课程的内容。请在机器上安装 MATLAB 程序，在提示符下输入 demo，运行演示程序，领略 MATLAB 语言的基本功能。

4 学会利用 MATLAB 语言提供的联机帮助功能，更好地学习和运用 MATLAB 语言，熟练掌握查找需要了解内容的方法和技巧。联机帮助可以由 help 命令或 Help 菜单来实现。试查找和 Lyapunov 关键词相关的函数名及调用方法，并求解下面给出的方程

$$\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}16&4&1\\9&3&1\\4&2&1\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$$

5 如果下载某 MATLAB 工具箱，则可以将其复制到某个文件夹，然后需要选择 File → Set Path 将该文件夹添加到 MATLAB 路径下。试下载某工具箱，如本书所附的 OCD 程序或 MATLAB Central 中下载的工具箱，将其添加到 MATLAB 路径下。

6 MATLAB 的 tic 和 toc 是一对计时的命令，前者启动秒表，后者可以读秒表。试执行命令 **A**=rand(500); tic, **B**=inv(**A**); toc，并从函数名称理解该代码所做的工作，由此体会 MATLAB 矩阵运算的速度。

7 用 MATLAB 语句输入矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$

$$\boldsymbol{A}=\begin{bmatrix}1&2&3&4\\4&3&2&1\\2&3&4&1\\3&2&4&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1+4\mathrm{j}&2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}\\4+1\mathrm{j}&3+2\mathrm{j}&2+3\mathrm{j}&1+4\mathrm{j}\\2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\\3+2\mathrm{j}&2+3\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\end{bmatrix}$$

前面给出的是 4×4 矩阵，如果给出 $\boldsymbol{A}(5,6)=5$ 命令将得出什么结果？

8 传统线性代数和数值分析教科书中介绍的矩阵特征值求解都是针对实数矩阵的，假若想求出上面的复数矩阵 $\boldsymbol{B}$ 的特征值，而不采用计算机数学语言，试搜索能求解这类问题的其他语言，如 C, FORTRAN 的子程序代码。

9 选择合适的步距绘制出图形 $\sin(1/t)$，其中 $t\in(-1,1)$。

10 用数值方法可以求出 $S=\sum_{i=0}^{63}2^i=1+2+4+8+\cdots+2^{62}+2^{63}$，试不采用循环的形式求出和式的数值解。由于数值方法是采用 double 形式进行计算的，难以保证有效位数字，所以结果不一定精确。试采用符号运算的方法求该和式的精确值。

参考文献

[1] Press W H, Flannery B P, Teukolsky S A, et al. Numerical recipes, the art of scientific computing [M]. Cambridge: Cambridge University Press, 1986

[2] Garbow B S, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide extension [M], Lecture notes in computer sciences, volume 51. New York: Springer-Verlag, 1977

[3] Smith B T, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide [M], Lecture notes in computer sciences, volume 6. New York: Springer-Verlag, second edition , 1976

[4] Dongarra J J, Bunsh J R, Molor C B. LINPACK user's guide [M]. Philadelphia: SIAM Press, 1979

[5] Numerical Algorithm Group. NAG FORTRAN library manual [Z], 1982

[6] Jobling C P, Grant P W, Barker H A, et al. Object-oriented programming in control system design: a survey [J]. Automatica, 1994, 30:1221~1261

[7] Melsa J L, Jones S K. Computer programs for computational assistance in the study of linear control theory [M]. New York: McGraw-Hill, 1973

[8] Åström K J. Computer aided tools for control system design. Jamshidi M, Herget C J, eds., Computer-aided control systems engineering, 3~40. Amsterdam: Elsevier Science Publishers B V, 1985

[9] Furuta K. Computer-aided design program for linear control systems [A]. Proceedings IFAC Symposium on CACSD [C]. Zurich,Switzerland, 1979, 267~272

[10] Edmunds J M. Cambridge linear analysis and design programs [A]. Proceedings IFAC Symposium on CACSD [C]. Zurich, Switzerland, 1979, 253~258

[11] Maciejowski J M, MacFarlane A G J. CLADP: the Cambridge linear analysis and design programs. Jamshidi M, Herget C J, eds., Computer-aided control systems engineering, 125~138. Amsterdam: Elsevier Science Publishers B V, 1985

[12] Armstrong E S. ORACLS — a design system for linear multivariable control [M]. New York: Marcel Dekker Inc., 1980

[13] Atherton D P. Control systems computed [J]. Physics in Technology, 1985, 16:139~140

[14] CAD Center. GINO-F Users' manual [Z], 1976

[15] Spang III H A. The federated computer-aided design system. Jamshidi M, Herget C J, eds., Computer-aided control systems engineering, 209~228. Amsterdam: Elsevier Science Publishers B V, 1985

[16] Munro N. ECSTASY — a control system CAD environment [A]. Proceedings IEE Conference on Control 88 [C]. Oxford, 1988, 76~80. Oxford

[17] Wolfram S. The Mathematica book [M]. Cambridge: Cambridge University Press, 1988

[18] Lamport L. LaTeX: a document preparation system — user's guide and reference manual [M]. Reading MA: Addision-Wesley Publishing Company, 2nd , 1994

[19] 王治宝，韩京清. CADCSC 软件系统 —— 控制系统计算机辅助设计 [M]. 北京: 科学出版社, 1997

[20] 孙增圻，袁曾任. 控制系统的计算机辅助设计 [M]. 北京: 清华大学出版社, 1988
[21] 吴重光、沈承林. 控制系统计算机辅助设计 [M]. 北京: 机械工业出版社, 1988
[22] Paláncz B, Benyó Z, Kovács L. Control system professional suite (Product Review) [J]. IEEE Control Systems Magazine, 2005, 25(4):67~75
[23] Benner P, Mehrmann V, Sima V, et al. SLICOT — A subroutine library in systems and control theory [R]. Technical Report 97-3, NICONET, 1997
[24] Kuo B C. Digital control systems [M]. New York: Holt, Rinehart and Winston. Inc, 1980
[25] Atherton D P. Nonlinear control engineering — describing function analysis and design [M]. London: Van Nostrand Reinhold, 1975
[26] Moler C B. MATLAB — An interactive matrix laboratory [R]. Technical Report 369, Department of Mathematics and Statistics, University of New Mexico, 1980
[27] 薛定宇，陈阳泉. 基于 MATLAB/Simulink 的系统仿真技术与应用 [M]. 北京: 清华大学出版社, 2002
[28] 张志涌. 精通 MATLAB 6.5 [M]. 北京: 北京航空航天大学出版社, 2003

第 2 章 微积分与积分变换的计算机求解

积分变换技术可以将某些难以分析的问题通过映射的方式映射到其他域内的表达式后再进行分析。例如，Laplace 变换可以将时域函数映射成复域函数，从而可以将某时域函数的微分方程映射成复域的多项式代数方程，使得原微分方程在诸多方面，如稳定性、解析解等方面更便于分析，这样的变换方法构成了经典自动控制理论的基础，引入了广泛采用的传递函数模型。

由于积分变换是定义在函数微积分的基础上的，所以本章 2.1 节先系统地介绍微积分问题的 MATLAB 求解，各种极限问题的求解方法，一般函数的微分计算，隐函数，参数方程以及多变量函数的微积分计算等，引入积分、广义积分、多重积分等问题的求解方法以及矩阵微积分问题及求解方法。本章 2.2 节首先介绍 Laplace 变换及其反变换的定义及性质，给出基于 MATLAB 的直接求解方法，并以一个简单例子介绍控制系统的微分方程建模方法，介绍基于 Laplace 变换的控制系统传递函数模型及其 MATLAB 表示，以及它在控制系统中的应用。2.3 节首先介绍 Fourier 级数的展开方法，Fourier 变换及其反变换的定义及性质，给出 MATLAB 语言下的 Fourier 变换问题求解方法，并将给出正余弦 Fourier 变换和离散 Fourier 正余弦变换问题的求解方法。2.4 节介绍 Z 变换及其反变换的定义与性质，基于 MATLAB 语言的求解方法，还将介绍基于 Z 变换的离散系统建模方法和连续、离散模型相互转换的方法。Laplace 变换和 Z 变换是整个控制理论中的重要基础，由这些变换构造出的连续和离散系统传递函数模型，所以可以围绕传递函数模型进行系统的分析与设计任务。2.5 节介绍有理函数的部分分式展开方法，引入留数计算的概念与求解方法，还将介绍部分分式展开在控制理论研究中的应用。综合这些知识，还将在 2.6 节中详细介绍控制系统模型的互联运算，控制系统结构图化简的方法及 MATLAB 实现方法。

2.1 微积分与矩阵微积分运算

在高等数学或数学分析课程中，介绍了各类函数的求导和积分运算，对各类不同类型函数的微积分运算读者需要记忆与识别，从而采用不同的策略来求解问题。本节将先给出求导运算的 MATLAB 求解，再针对多元函数偏导数、隐函数偏导数和矩阵求导几个特殊问题的计算机求解进行探讨。最后给出基于 MATLAB 符号运算的积分运算的统一方法。

2.1.1 极限问题的解析解

1. 单变量函数的极限

假设已知函数 $f(x)$，则极限问题的一般描述为

$$L=\lim_{x\to x_0} f(x) \tag{2-1-1}$$

其中，x_0 可以是一个确定的值，也可以是无穷大，例如 $x\to\infty$。对某些函数来说，还可以如下定义单边极限 (或称左右极限) 问题。

$$L_1=\lim_{x\to x_0^-} f(x) \text{ 或 } L_2=\lim_{x\to x_0^+} f(x) \tag{2-1-2}$$

前者表示 x 从左侧趋近于 x_0 点，所以又称为左极限，后者相应地称为右极限。极限问题在 MATLAB 符号运算工具箱中可以使用 `limit()` 函数直接求出，该函数的调用格式为

```
L=limit(fun, x, x0)                     % 求极限
L=limit(fun, x, x0, 'left' 或 'right')  % 求单边极限
```

在求解之前应该先申明自变量 x，再定义极限表达式 `fun`，若 x_0 为 ∞，则可以用 `inf` 直接表示。如果需要求解左右极限问题，还需要给出左右选项。调用此函数直接求解问题的好处在于，用户无需再像学习高等数学那样去记忆和掌握各种特殊极限问题的标准型，再套用公式求解问题了，有这个函数可以直接得出所需的极限值。下面将通过例子演示 MATLAB 求解极限的方法。

例 2-1 试求解极限问题 $\displaystyle\lim_{x\to\infty}\frac{(x+2)^{x+2}(x+3)^{x+3}}{(x+5)^{2x+5}}\left(\sqrt{x+x^2}-\sqrt{x^2-x}\right)$。

求解 利用 MATLAB 语言，应该首先申明 x 为符号变量，然后定义极限式子，最后调用 `limit()` 函数求出给定函数的极限，结果为 e^{-5}。

```
>> syms x; f=(x+2)^(x+2)*(x+3)^(x+3)/(x+5)^(2*x+5)...
      *(sqrt(x+x^2)-sqrt(x^2-x)); L=limit(f,x,inf)
```

例 2-2 试求解单边极限问题 $\lim\limits_{x\to 0^+}\dfrac{\mathrm{e}^{x^3}-1}{1-\cos\sqrt{x-\sin x}}$。

求解 利用 MATLAB 语言的 limit() 函数，可以容易地求取单边极限。

```
>> syms x;
   limit((exp(x^3)-1)/(1-cos(sqrt(x-sin(x)))),x,0,'right')
```

用单边极限的方法可求出该函数的极限值为 12。

回顾原始问题，其中采用 $x\to 0^+$ 是因为它可以保证根号内的值为非负数。事实上，即使是负数，$\cos(\mathrm{j}\alpha)=(\mathrm{e}^{\alpha}+\mathrm{e}^{-\alpha})/2$ 也是有定义的，故对这个问题没有影响。但对分段函数来说，单边极限是有意义的。

2. 多变量函数的极限

多元函数的极限也可以同样用 MATLAB 中的 limit() 函数直接求解。假设有二元函数 $f(x,y)$，若想求出二元函数的极限

$$L=\lim_{\substack{x\to x_0\\ y\to y_0}} f(x,y) \tag{2-1-3}$$

则可以嵌套使用 limit() 函数。例如，

L_1=limit(limit(f,x,x_0),y,y_0) 或

L_1=limit(limit(f,y,y_0),x,x_0)

如果 x_0 或 y_0 不是确定的值，而是另一个变量的函数，例如 $x\to g(y)$，则上述的极限求取顺序不能交换。

例 2-3 试求出二元函数极限值 $\lim\limits_{\substack{x\to 1/\sqrt{y}\\ y\to\infty}} \mathrm{e}^{-1/(y^2+x^2)}\dfrac{\sin^2 x}{x^2}\left(1+\dfrac{1}{y^2}\right)^{x+a^2y^2}$。

求解 本例中的问题可以用下面的语句直接解出，结果为 e^{a^2}。

```
>> syms x y a;
   f=exp(-1/(y^2+x^2))*sin(x)^2/x^2*(1+1/y^2)^(x+a^2*y^2);
   L=limit(limit(f,x,1/sqrt(y)),y,inf)
```

2.1.2 微分运算的 MATLAB 求解

MATLAB 语言直接提供了函数的高阶导数求导函数 diff()，但对于一些特殊的求导问题，如参数方程的求导问题和隐函数求导问题等，则需要根据其数学定义由 MATLAB 现有的语句组合或编写相应的函数来求得，这里将分各种情况介绍各种微分运算求解的问题。

1. 函数的导数和高阶导数

如果函数和自变量都已知，且均为符号变量，则可以用 diff() 函数解出给定函数的各阶导数。diff() 函数的调用格式为

```
y=diff(fun,x)     % 求导数
y=diff(fun,x,n)   % 求 n 阶导数
```

其中，fun 为给定函数，x 为自变量，这两个变量均应该为符号型的，n 为导数的阶次，若省略 n 则将自动求取一阶导数。

例 2-4 给定函数 $f(x)=\dfrac{\mathrm{e}^{-5x}\sin 2x}{x^2+3x+5}$，试求出 $\dfrac{\mathrm{d}^3f(x)}{\mathrm{d}x^3}$。

求解 函数的高阶导数问题可以很容易地用 MATLAB 符号运算工具箱语句求解。可以首先申明 x 为符号变量，然后调用 diff() 函数就能直接得出函数的三阶导数。

```
>> syms x; f=exp(-5*x)*sin(2*x)/(x^2+3*x+5); f1=diff(f,x,3)
```

这时可以得出函数的三阶导数为

$$-65\frac{\mathrm{e}^{-5x}\sin 2x}{x^2+3x+5}+142\frac{\mathrm{e}^{-5x}\cos 2x}{x^2+3x+5}-63\frac{\mathrm{e}^{-5x}(2x+3)\sin 2x}{(x^2+3x+5)^2}+12\frac{\mathrm{e}^{-5x}(2x+3)^2\cos 2x}{(x^2+3x+5)^3}$$
$$+60\frac{\mathrm{e}^{-5x}(2x+3)\cos 2x}{(x^2+3x+5)^2}-30\frac{\mathrm{e}^{-5x}(2x+3)^2\sin 2x}{(x^2+3x+5)^3}+30\frac{\mathrm{e}^{-5x}\sin 2x}{(x^2+3x+5)^2}$$
$$-12\frac{\mathrm{e}^{-5x}\cos 2x}{(x^2+3x+5)^2}-6\frac{\mathrm{e}^{-5x}(2x+3)^3\sin 2x}{(x^2+3x+5)^4}+12\frac{\mathrm{e}^{-5x}(2x+3)\sin 2x}{(x^2+3x+5)^3}$$

由原函数和得出的三阶导数函数可以对给定自变量向量 x 求出各点处的函数值，并由下面的 MATLAB 命令绘制出来，如图 2-1 所示。从图中可以看出原函数和其三

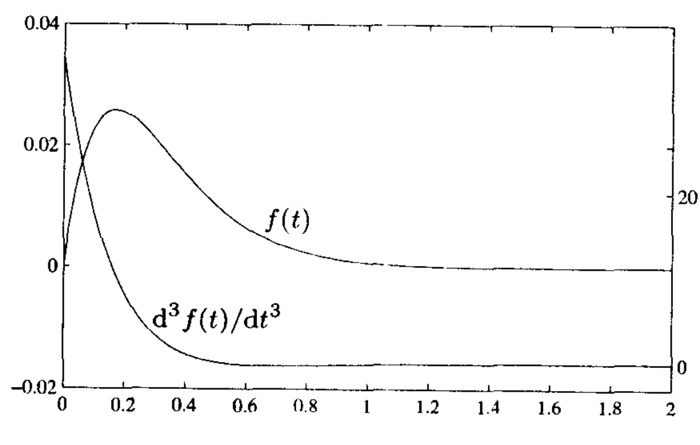


图 2-1　函数及其三阶导数图示

阶导数之间的关系。注意由于采用了 plotyy()，故得出的是双纵坐标的图形，因为两个图幅值相差较悬殊。

```
>> x1=0:.01:2; y=subs(f,x,x1); y1=subs(f1,x,x1);
   plotyy(x1,y,x1,y1)     % 原函数及其三阶导数
```

MATLAB 现成的 diff() 函数还适合于求解给定函数更高阶的导数。例如，下面

给出的命令可以在 8.5 秒内获得函数的 100 阶导函数。

```
>> tic, diff(f,x,100); toc
```

2. 多元函数的偏导数

MATLAB 的符号运算工具箱中并未提供求取偏导数的专门函数，这些偏导数仍然可以通过 diff() 函数直接实现。假设已知二元函数 $f(x,y)$，若想求 $\dfrac{\partial^{m+n} f}{\partial x^m \partial y^n}$，则可以用下面的函数求出。

f_1=diff(diff(f,x,m),y,n)　或

f_2=diff(diff(f,y,n),x,m)

例 2-5 试求出二元函数 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$ 的偏导数，并用图形表示。

求解 用下面的语句可直接求出 $\dfrac{\partial z}{\partial x}$ 与 $\dfrac{\partial z}{\partial y}$。

```
>> syms x y; z=(x^2-2*x)*exp(-x^2-y^2-x*y);
   zx=simple(diff(z,x)), zy=diff(z,y)
```

其数学形式分别为

$$\frac{\partial z(x,y)}{\partial x}=-\mathrm{e}^{-x^2-y^2-xy}(-2x+2+2x^3+x^2y-4x^2-2xy)$$
$$\frac{\partial z(x,y)}{\partial y}=-x(x-2)(2y+x)\mathrm{e}^{-x^2-y^2-xy}$$

在 $x\in(-3,3), y\in(-2,2)$ 区域内生成网格，则可以分别得出原函数及其偏导数的数值解。这样，可以直接用下面语句绘制出原函数的三维曲面，如图 2-2 (a) 所示。

```
>> [x1,y1]=meshgrid(-3:.2:3,-2:.2:2); z1=subs(z,{x,y},{x1,y1});
   surf(x1,y1,z1), axis([-3 3 -2 2 -0.7 1.5]) % 直接绘制三维曲面
```

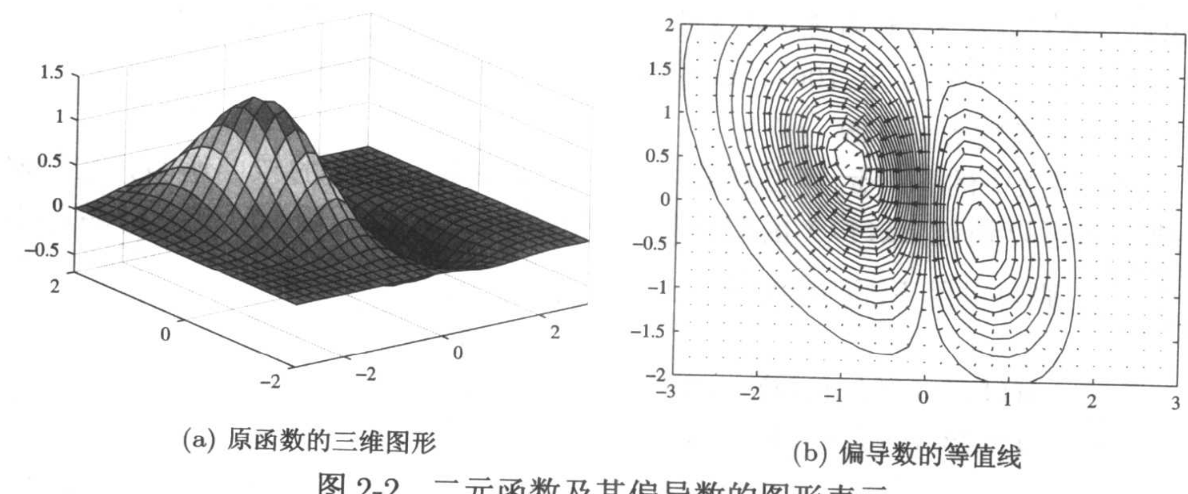
既然计算出了对两个自变量的一阶偏导数，则可以调用 quiver() 函数绘制出引力线，该引力线可以叠印在用 contour() 函数绘制出的等值线上，如图 2-2 (b) 所示。其中，引力线为对应点处的梯度向量，其绘制函数的详细信息可以由 help quiver 命令进一步列出。

```
>> contour(x1,y1,z1,30), hold on    % 绘制等值线
   zx1=subs(zx,{x,y},{x1,y1}); zy1=subs(zy,{x,y},{x1,y1});
   quiver(x1,y1,zx1,zy1)  % 绘制引力线
```

例 2-6 已知三元函数 $f(x,y,z)=\sin(x^2y)\mathrm{e}^{-x^2y-z^2}$，试求出偏导数 $\dfrac{\partial^4 f(x,y,z)}{\partial x^2\partial y\partial z}$。

求解 由下面的语句申明自变量及函数，则可以用 MATLAB 语句立即得出所需的偏导函数。

```
>> syms x y z; f=sin(x^2*y)*exp(-x^2*y-z^2);
```



(a) 原函数的三维图形

(b) 偏导数的等值线

图 2-2 二元函数及其偏导数的图形表示

```
df=diff(diff(diff(f,x,2),y),z); df=simple(df); latex(df);
```

可以得出

$$-4ze^{-x^2y-z^2}\left[\cos\left(x^2y\right)-10\cos\left(x^2y\right)yx^2+4x^4\sin\left(x^2y\right)y^2+4\cos\left(x^2y\right)x^4y^2-\sin\left(x^2y\right)\right]$$

3. 隐函数的偏导数

已知隐函数的数学表达式为 $f(x_1, x_2, \cdots, x_n) = 0$，则可以通过隐函数对它们的偏导数求出自变量之间的偏导数。具体可以用下面的公式求出 $\dfrac{\partial x_i}{\partial x_j}$。

$$\frac{\partial x_i}{\partial x_j} = -\frac{\dfrac{\partial}{\partial x_j} f(x_1, x_2, \cdots, x_n)}{\dfrac{\partial}{\partial x_i} f(x_1, x_2, \cdots, x_n)} \tag{2-1-4}$$

由于 f 对 x_i, x_j 的偏导数可以分别由 diff() 函数求出，故整个偏导数可以由它们的除法获得，所以这样的问题可以用 MATLAB 语句容易地得出。

```
F=-diff(f,xj)/diff(f,xi)
```

例 2-7 考虑二元函数 $f(x,y) = x^3 + y^2 \sin x - y - 6 = 0$，试求 $\dfrac{\partial y}{\partial x}$。

求解 根据式 (2-1-4) 可以立即得出所需偏导数 $\dfrac{\partial y}{\partial x}$ 为

```
>> syms x y; f=x^3+y^2*sin(x)-y-6; simple(-diff(f,x)/diff(f,y))
```

这样可以得出结果为 $-\dfrac{3x^2 + y^2\cos x}{2y\sin x - 1}$。

4. 参数方程的导数

若已知参数方程 $y = f(t), x = g(t)$，则 $\dfrac{d^n y}{dx^n}$ 可以由下面的 MATLAB 语句递推求出。

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x}=\dfrac{f'(t)}{g'(t)} \\ \dfrac{\mathrm{d}^2y}{\mathrm{d}x^2}=\dfrac{\mathrm{d}}{\mathrm{d}t}\left(\dfrac{f'(t)}{g'(t)}\right)\dfrac{1}{g'(t)}=\dfrac{\mathrm{d}}{\mathrm{d}t}\left(\dfrac{\mathrm{d}y}{\mathrm{d}x}\right)\dfrac{1}{g'(t)} \\ \qquad\vdots \\ \dfrac{\mathrm{d}^ny}{\mathrm{d}x^n}=\dfrac{\mathrm{d}}{\mathrm{d}t}\left(\dfrac{\mathrm{d}^{n-1}y}{\mathrm{d}x^{n-1}}\right)\dfrac{1}{g'(t)} \end{cases} \tag{2-1-5}$$

应用递归调用方式，可以编写出如下的 MATLAB 函数求解偏导数

```
function result=paradiff(y,x,t,n)
if mod(n,1)~=0, error('Error: n should positive integer');
else
   if  n==1, result=diff(y,t)/diff(x,t);
   else, result=diff(paradiff(y,x,t,n-1),t)/diff(x,t);
end, end
```

例 2-8 已知参数方程 $y=\dfrac{3at^2}{1+t^3}$，$x=\dfrac{3at}{1+t^3}$，试求 $\dfrac{\mathrm{d}y}{\mathrm{d}x}$，$\dfrac{\mathrm{d}^3y}{\mathrm{d}x^3}$，$\dfrac{\mathrm{d}^5y}{\mathrm{d}x^5}$。

求解 由前面给出的函数调用格式，可以立即得出所需的高阶导数。

```
>> syms t a;
   x=3*a*t/(1+t^3); y=3*a*t^2/(1+t^3); y1=paradiff(y,x,t,1)
   y3=simple(paradiff(y,x,t,3)), y5=simple(paradiff(y,x,t,5))
```

可以自动得出如下的结果。

$$\frac{\mathrm{d}y}{\mathrm{d}x}=\frac{t\left(-2+t^3\right)}{-1+2t^3}$$

$$\frac{\mathrm{d}^3y}{\mathrm{d}x^3}=\frac{4}{3}\frac{\left(-5+t^3\right)(t+1)^5\left(t^2-t+1\right)^5t^2}{a^2\left(-1+2t^3\right)^5}$$

$$\frac{\mathrm{d}^5y}{\mathrm{d}x^5}=\frac{40}{27}\frac{\left(14t^{15}-98t^{12}+350t^9-1138t^6-143t^3-1\right)(t+1)^7\left(t^2-t+1\right)^7}{a^4\left(-1+2t^3\right)^9}$$

2.1.3 积分运算

不定积分、定积分以及多重积分的数学表达式可以分别写成

$$\int f(x)\mathrm{d}x,\quad \int_a^b f(x)\mathrm{d}x,\quad \int\cdots\int f(x_1,x_2,\cdots,x_n)\mathrm{d}x_n\cdots\mathrm{d}x_2\mathrm{d}x_1 \tag{2-1-6}$$

给定被积函数 $f(\cdot)$，则按照传统高等数学求解方法，需要人工识别其函数类型并选择适当的积分方法，如分部积分法、换元积分法等合适方法求解原问题，这在很

大程度上取决于经验。MATLAB 符号运算工具箱中提供了一个 int() 函数，可以直接用来求取符号函数的不定积分。该函数无须人工识别所需积分算法，可以理解成一种统一的积分问题求解方法。该函数的调用格式为

```
F=int(fun,x)        % 不定积分运算
F=int(fun,x,a,b)    % 定积分运算，其中 a,b 可以不是常数
```

如果被积函数 fun 中只有一个变量，则调用语句中的 x 可以省略。另外，该函数得出的结果 F 是积分原函数，实际的不定积分应该是 $F+C$ 构成的函数族。其中，C 是任意常数。int() 函数还可以用于无穷积分限的广义积分计算，如用 inf 和 $-$inf 分别表示 ∞ 和 $-\infty$。

对于可积的函数，MATLAB 符号运算工具箱提供的 int() 函数可以用计算机代替繁重的手工推导，立即得出原始问题的解。而对于不可积的函数来说，当然 MATLAB 也是无能为力的。

多重积分可以由 int() 函数嵌套求出。

下面将通过例子介绍 int() 使用方法及其在积分问题求解中的应用。

例 2-9 考虑例 2-4 中给出的问题，用 diff() 函数可以直接求 $f(x)$ 函数的一阶导数。现在对得出的导数再进行积分，试检验是否可以得出一致的结果。

求解 先定义原函数并对其求导，然后再对导数进行积分，则

```
>> syms x; y=sin(x)/(x^2+4*x+3); y1=diff(y);    % 对函数求导
   y0=int(y1); latex(y0)      % 对导数积分应该得出原函数
```

得出的结果为 $1/2\dfrac{\sin x}{x+1}-1/2\dfrac{\sin(x)}{x+3}$。现在对原函数求三阶导数，再对结果进行 3 次积分，则可以用下面语句判定正确性。

```
>> y3=diff(y,3); y0=int(int(int(y3))); latex(simple(y0))
```

得出的结果为 $\dfrac{\sin x}{(x+1)(x+3)}$，其结果和原函数一样。

例 2-10 试研究解析不可积函数 $f(x)=\mathrm{e}^{-x^2/2}$ 的积分问题。

求解 直接采用 MATLAB 的 int() 函数计算 $\displaystyle\int_3^5 f(x)\mathrm{d}x$

```
>> syms x; F=int(exp(-x^2/2),x,3,5)
```

则可以得出 $F=\dfrac{\sqrt{2\pi}}{2}\mathrm{erf}\left(\dfrac{5}{2}\sqrt{2}\right)-\dfrac{\sqrt{2\pi}}{2}\mathrm{erf}\left(\dfrac{3}{2}\sqrt{2}\right)$，其中 MATLAB 语言使用了数学上定义的函数 $\mathrm{erf}(x)=\dfrac{2}{\pi}\displaystyle\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$，故可以得出原问题的解析解。然而这样的解毕竟无法用于工程实际，所以应该采用 vpa() 函数求取其高精度的数值解，由 vpa(F) 命令可以得出 0.0033829740450176877779095235876。

例 2-11 考虑下面的三元函数 $F(x,y,z)=-4z\mathrm{e}^{-x^2y-z^2}\left[\cos\left(x^2y\right)-10\cos\left(x^2y\right)yx^2\right.$ $\left.+4x^4\sin\left(x^2y\right)y^2+4\cos\left(x^2y\right)x^4y^2-\sin\left(x^2y\right)\right]$，试求出 $\int\cdots\int F(x,y,z)\mathrm{d}x^2\mathrm{d}y\mathrm{d}z$。

求解 事实上，此函数是例 2-6 中给出 $f(x,y,z)$ 经偏导运算得出，故需要对求导过程进行逆向运算，还原回原函数的结果。

对该函数进行积分。先对 z 积分一次，对 y 积分一次，再连续对 x 积分两次，经过化简，则得出如下结果。

```
>> syms x y z;
   f0=-4*z*exp(-x^2*y-z^2)*(cos(x^2*y)-10*cos(x^2*y)*y*x^2+...
      4*sin(x^2*y)*x^4*y^2+4*cos(x^2*y)*x^4*y^2-sin(x^2*y));
   f1=int(f0,z); f1=int(f1,y); f1=int(f1,x); f1=simple(int(f1,x))
```

则可以得出 $f_1=\sin(x^2y)\mathrm{e}^{-x^2y-z^2}$。可见，经化简按照上述给出的积分顺序可以得出原函数，和例 2-6 中给出的原函数完全一致。现在考虑改变积分求解顺序，例如，将积分顺序变成 $z\rightarrow x\rightarrow x\rightarrow y$，则可以得出如下结果。

```
>> f2=int(f0,z); f2=int(f2,x); f2=int(f2,x); f2=simple(int(f2,y))
```

这时得出的积分结果为 $f_2=\dfrac{2\mathrm{e}^{-x^2y-z^2}\tan\left(x^2y/2\right)}{1+\tan^2\left(x^2y/2\right)}$。

可见，得出的化简结果显然不同于原函数。由 simple(f1-f2) 语句将其和原函数理论值相减，则可以得出误差为 0，表明二者实际上是完全一致的，但由于积分顺序选择不同，得不出实际的最简形式。

2.2 Laplace 变换及反变换

法国数学家 Pierre-Simon Laplace (1749–1827) 引入的积分变换可以巧妙地将一般常系数微分方程映射成代数方程，奠定了很多领域，如电路分析、自动控制原理等的数学模型基础。本节将首先介绍 Laplace 变换及反变换的定义与性质，然后介绍利用 MATLAB 求解 Laplace 变换及反变换的方法与应用。

2.2.1 Laplace 变换及反变换定义与性质

一个时域函数 $f(t)$ 的 Laplace 变换可以定义为

$$\mathscr{L}[f(t)]=\int_0^{\infty}f(t)\mathrm{e}^{-st}\mathrm{d}t=F(s) \tag{2-2-1}$$

式中，$\mathscr{L}[f(t)]$ 为 Laplace 变换的简单记号。

下面将不加证明地列出一些 Laplace 变换的性质。

① **线性性质**　若 a 与 b 均为标量，则 $\mathscr{L}[af(t)\pm bg(t)]=a\mathscr{L}[f(t)]\pm b\mathscr{L}[g(t)]$。

② **时域平移性质**　$\mathscr{L}[f(t-a)]=\mathrm{e}^{-as}F(s)$。

③ s **域平移性质**　$\mathscr{L}[\mathrm{e}^{-at}f(t)]=F(s+a)$。

④ **微分性质**　$\mathscr{L}[\mathrm{d}^n f(t)/\mathrm{d}t^n]=sF(s)-f(0^+)$，一般地，对 n 阶微分有

$$\mathscr{L}\left[\frac{\mathrm{d}^n}{\mathrm{d}t^n}f(t)\right]=s^nF(s)-s^{n-1}f(0^+)-s^{n-2}\frac{\mathrm{d}f(0^+)}{\mathrm{d}t}-\cdots-\frac{\mathrm{d}^{n-1}f(0^+)}{\mathrm{d}t^{n-1}} \tag{2-2-2}$$

若假设函数 $f(t)$ 及其各阶导数的初值均为 0，则式 (2-2-2) 可以简化成

$$\mathscr{L}\left[\frac{\mathrm{d}^n f(t)}{\mathrm{d}t^n}\right]=s^nF(s) \tag{2-2-3}$$

此性质事实上是微分方程映射成代数方程的关键式子。

⑤ **积分性质**　若假设零初始条件，$\mathscr{L}\left[\int_0^t f(\tau)\mathrm{d}\tau\right]=\dfrac{F(s)}{s}$，一般地，函数 $f(t)$ 的 n 重积分的 Laplace 变换可以由下式求出。

$$\mathscr{L}\left[\int_0^t\cdots\int_0^t f(\tau)(\mathrm{d}\tau)^n\right]=\frac{F(s)}{s^n} \tag{2-2-4}$$

⑥ **初值性质**　$\lim\limits_{t\to 0}f(t)=\lim\limits_{s\to\infty}sF(s)$。

⑦ **终值性质**　如果 $F(s)$ 没有 $s\geqslant 0$ 的极点，则 $\lim\limits_{t\to\infty}f(t)=\lim\limits_{s\to 0}sF(s)$。

⑧ **卷积性质**　$\mathscr{L}[f(t)*g(t)]=\mathscr{L}[f(t)]\mathscr{L}[g(t)]$，式中，卷积算子 $*$ 的定义为

$$f(t)*g(t)=\int_0^t f(\tau)g(t-\tau)\mathrm{d}\tau=\int_0^t f(t-\tau)g(\tau)\mathrm{d}\tau \tag{2-2-5}$$

⑨ **其他性质**

$$\mathscr{L}[t^nf(t)]=(-1)^n\frac{\mathrm{d}^nF(s)}{\mathrm{d}s^n},\quad \mathscr{L}\left[\frac{f(t)}{t^n}\right]=\int_s^\infty\cdots\int_s^\infty F(s)\mathrm{d}s^n \tag{2-2-6}$$

如果已知函数的 Laplace 变换式 $F(s)$，则可以通过下面的反变换公式求出其 Laplace 反变换

$$f(t)=\mathscr{L}^{-1}[F(s)]=\frac{1}{2\mathrm{j}\pi}\int_{\sigma-\mathrm{j}\infty}^{\sigma+\mathrm{j}\infty}F(s)\mathrm{e}^{st}\mathrm{d}s \tag{2-2-7}$$

其中 σ 大于 $F(s)$ 奇点的实部，奇点的概念将在后面给出。

2.2.2 Laplace 变换的计算机求解

求解已知函数的 Laplace 变换用数值编程的方式是无法实现的，如果需要采用计算机来求解这样的问题，必须借助于计算机数学语言，如本书介绍的 MATLAB 语言。MATLAB 语言的符号运算工具箱可以轻松地求解 Laplace 变换问题。具体的变换及反变换问题的求解步骤如下：

① 定义符号变量 t，这样就能描述时域表达式 Fun 了。和前面介绍的内容一样，申明符号变量可以用 syms 命令实现。

② 直接调用 laplace() 函数，就可以求解出所需的时域函数 Laplace 变换式子。该函数的调用格式为

```
F=laplace(Fun)        % 采用默认的 t 为时域变量
F=laplace(Fun,v,u)    % 用户指定时域变量 v 和复域变量名 u
```

还可以考虑采用 simple() 等函数对其进行化简。

③ 对复杂的问题来说，得出的结果形式通常难以阅读，所以需要调用符号运算工具箱的 pretty() 函数或 latex() 函数对结果进行进一步处理。可以在屏幕上或利用 LaTeX 的强大功能将结果用可读性更强的形式显示出来。

如果已知 Laplace 变换式，则应该首先给出 Laplace 变换式子 Fun，然后采用符号运算工具箱中的 ilaplace() 函数对其进行反变换。该函数的调用格式为

```
f=ilaplace(Fun)       % 采用默认的 s 为时域变量
f=ilaplace(Fun,u,v)   % 用户指定时域变量 v 和复域变量名 u
```

获得变换式子之后也可以对之进一步化简和改变显示格式。

例 2-12 已知函数 $f(t)=te^{-3t}\sin 2t$，试求取该函数的 Laplace 变换。

求解 分析原题，可以先申明 t 为符号变量，再用 MATLAB 语句表示给定的 $f(t)$ 函数，然后就可以用下面的语句立即得出该函数的 Laplace 变换为

```
>> syms t; f=t*exp(-3*t)*sin(2*t); laplace(f)
```

由前面的语句可以直接得出变换结果 $\mathscr{L}[y(t)]=\dfrac{s+3}{4\left(1/4\,(s+3)^2+1\right)^2}$。

例 2-13 试求出 $Y(s)=\dfrac{s^3+7s^2+24s+24}{s(s^4+10s^3+35s^2+50s+24)}$ 的 Laplace 反变换。

求解 不难看出，该反变换实际上就是系统 $G(s)=\dfrac{s^3+7s^2+24s+24}{s^4+10s^3+35s^2+50s+24}$ 的阶跃响应解析解。下面的语句可以直接求解该问题。

```
>> syms s; Y=(s^3+7*s^2+24*s+24)/s/(s^4+10*s^3+35*s^2+50*s+24);
   ilaplace(Y)
```

这样可以得出反变换的结果为 $y(t)=\mathscr{L}^{-1}[Y(s)]=1-\mathrm{e}^{-2t}-\mathrm{e}^{-t}-\mathrm{e}^{-4t}+2\mathrm{e}^{-3t}$。

例 2-14 试求出下面函数的 Laplace 反变换。

$$G(x)=\frac{-17x^5-7x^4+2x^3+x^2-x+1}{x^6+11x^5+48x^4+106x^3+125x^2+75x+17}$$

求解 从原来给出的问题，似乎用下面的语句就能直接求解出所需结果。

```
>> syms x t; G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)...
     /(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17);
   f=ilaplace(G,x,t)
```

得出的结果为

```
-1/31709*sum((39275165+45806941*_alpha^4+156459285*_alpha+
5086418*_alpha^5+149142273*_alpha^3+221566216*_alpha^2)*exp(_alpha*t),
_alpha = RootOf(_Z^6+11*_Z^5+48*_Z^4+106*_Z^3+125*_Z^2+75*_Z+17))
```

而事实上，该方法并不能求出有意义的结果。这是因为分母多项式对应的方程的解不存在解析解，所以导致原问题不存在解析解。若利用 MATLAB 的变精度算法 `simple(vpa(f,16))`，则可以得出高精度的数值解。因篇幅所限，这里只能给出其近似结果，即

$$-556.256\mathrm{e}^{-3.2617t}+537.286\mathrm{e}^{-2.5309t}\cos(0.39976t)-698.247\mathrm{e}^{-2.5309t}\sin(0.39976t)$$
$$+1.75893\mathrm{e}^{-1.07776t}\cos(0.6021t)+10.9942\mathrm{e}^{-1.07776t}\sin(0.6021t)+0.2126\mathrm{e}^{-0.52t}$$

注意，由于这里涉及高阶多项式方程的求解，而该方程本身并没有解析解，所以这个问题也只能求出高精度的数值解，但无法求出解析解，因为解析解根本不存在。

例 2-15 对例 2-12 给出的原函数 $f(t)$，试研究 $\mathscr{L}[\mathrm{d}^5f(t)/\mathrm{d}t^5]$ 和 $s^5\mathscr{L}[f(t)]$ 的关系。

求解 如果想求解这样的问题，可以利用符号运算工具箱中的 `diff()` 函数对函数 $f(t)$ 求五阶导数，再进行 Laplace 变换，则

```
>> syms t; f=t*exp(-3*t)*sin(2*t); F=simple(laplace(diff(f,t,5)))
```

可以得出 Laplace 变换为 $-4\dfrac{1617s^2+5655s+7774+150s^3}{(s^2+6s+13)^2}$。

对 $f(t)$ 进行 Laplace 变换，并将变换结果乘以 s^5，将得出的结果与前面直接得出的结果相减，则

```
>> syms s; F0=laplace(f); simple(F-s^5*F0)
```

这样得出二者之差为 $-4s^2+36s-184$。由于二者之差不为 0，所以看起来和式 (2-2-3) 是不同的。这是因为 $f(t)$ 函数有 $f(0)=0$，但其各阶导数在 $t=0$ 处的值不为 0，故不满足式 (2-2-3)，而满足式 (2-2-2)。

例 2-16 试推导出 $\mathscr{L}\left[\dfrac{\mathrm{d}^8f(t)}{\mathrm{d}t^8}\right]$ 的微分公式。

求解 MATLAB 的符号运算工具箱还可以进行一些简单的 Laplace 变换公式推导。假设想导出 $f(t)$ 的 8 阶导数的 Laplace 变换，首先应该先定义一下 $f(t)$ 函数，这可以通过如下语句实现，并推导出 8 阶导数的 Laplace 变换公式。

```
>> syms t; y=sym('f(t)'); laplace(diff(y,t,8))
```

这样将得出如下的结果，这和式 (2-2-2) 中的结论是完全一致的。

$s(s(s(s(s(s(sF(s)-f(0))-\dot{f}(0))-\ddot{f}(0))-f^{(3)}(0))-f^{(4)}(0))-f^{(5)}(0))-f^{(6)}(0))-f^{(7)}(0)$

2.2.3 控制系统的传递函数模型

利用 Laplace 变换的微分性质就可以将难以处理的微分方程变换成代数方程，从而可以建立起系统的传递函数模型。本节首先通过一个简单的串联 RLC 电路来演示实际系统的微分方程建模，引入连续系统的传递函数与零极点模型表示，介绍这些模型直接的相互转换方法，然后通过 Laplace 反变换的方法得出典型二阶系统阶跃响应解析解的简洁表示方式。

1. 控制系统的数学模型

这里以一个实际系统为例来介绍系统模型的推导。考虑图 2-3 中给出电阻 R，电感 L 和电容 C 以串联形式连接起来的电路图。电流 $i(t)$ 满足方程

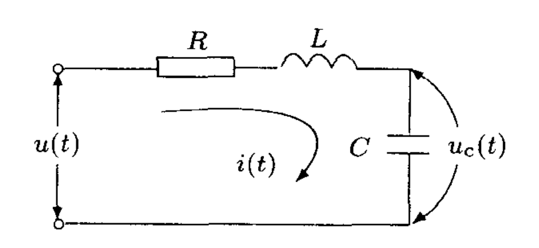


图 2-3 RLC 串联电路

$$i(t)=C\frac{\mathrm{d}u_{\mathrm{c}}(t)}{\mathrm{d}t} \tag{2-2-8}$$

同时，可以写出电容元件两端的电压 $u_{\mathrm{c}}(t)$ 的方程表示为

$$u(t)=Ri(t)+L\frac{\mathrm{d}i(t)}{\mathrm{d}t}+u_{\mathrm{c}}(t) \tag{2-2-9}$$

从这两个方程可以写出

$$LC\frac{\mathrm{d}^2u_{\mathrm{c}}(t)}{\mathrm{d}t^2}+RC\frac{\mathrm{d}u_{\mathrm{c}}(s)}{\mathrm{d}t}+u_{\mathrm{c}}(t)=u(t) \tag{2-2-10}$$

这时式 (2-2-10) 为一个二阶常微分方程，这个方程称为该电路的数学模型，其中 $u(t)$ 为系统的输入信号，$u_{\mathrm{c}}(t)$ 为输出信号。连续动态系统的数学模型一般都可以表示成常微分方程的形式。更一般地，假设系统的输入信号为 $u(t)$，且输出信号为 $y(t)$，则线性系统的微分方程可以写成

$$\begin{aligned}&a_1\frac{\mathrm{d}^ny(t)}{\mathrm{d}t^n}+a_2\frac{\mathrm{d}^{n-1}y(t)}{\mathrm{d}t^{n-1}}+\cdots+a_n\frac{\mathrm{d}y(t)}{\mathrm{d}t}+a_{n+1}y(t)\\&=b_1\frac{\mathrm{d}^mu(t)}{\mathrm{d}t^m}+b_2\frac{\mathrm{d}^{m-1}u(t)}{\mathrm{d}t^{m-1}}+\cdots+b_m\frac{\mathrm{d}u(t)}{\mathrm{d}t}+b_{m+1}u(t)\end{aligned} \tag{2-2-11}$$

其中 n 又称为系统的阶次。

2. 控制系统的传递函数模型

由于很久以前并没有微分方程的实用求解工具，所以可以利用 Laplace 变换，对微分方程进行变换，将其映射成代数方程来求解。假设 $y(t)$ 信号的 Laplace 变换式为 $Y(s)$，并假设该信号及其各阶导数的初始值均为 0，将 Laplace 变换的重要性质 $\mathscr{L}[\mathrm{d}^k y(t)/\mathrm{d}t^k] = s^k Y(s)$ 代入式 (2-2-11) 中给出的微分方程，则可以巧妙地将微分方程映射成多项式方程。输出信号和输入信号 Laplace 变换的比值定义为增益，该比值又称为系统的传递函数，从变换后得出的多项式方程可以立即得出单变量连续线性系统的传递函数为

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_1 s^m + b_2 s^{m-1} + \cdots + b_m s + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \cdots + a_n s + a_{n+1}} \tag{2-2-12}$$

其中 b_i $(i = 1, \cdots, m+1)$ 与 a_i $(i = 1, \cdots, n+1)$ 为常数。这样的系统又称为线性时不变 (linear time invariant，LTI，又称为线性定常) 系统。系统的分母多项式又称为系统的特征多项式。对物理可实现系统来说，一定要满足 $m \leqslant n$，这种情况下又称系统为正则 (proper) 系统。若 $m < n$，则称系统为严格正则。$n - m$ 又称为系统的相对阶次。

从式 (2-2-12) 中可以看出，传递函数可以表示成两个多项式的比值，在 MATLAB 语言中，多项式可以用向量表示。依照 MATLAB 惯例，将多项式的系数按 s 的降幂次序表示就可以得到一个数值向量，用这个向量就可以表示多项式。分别表示分子和分母多项式后，再利用控制系统工具箱的 `tf()` 函数就可以用一个变量表示传递函数变量 G：

```
num=[b1,b2,…,bm,bm+1];    den=[a1,a2,…,an,an+1];
G=tf(num,den);
```

MATLAB 还支持另一种特殊的传递函数的输入格式，在这样的输入方式下，应该用 `s=tf('s')` 先定义传递函数的算子，然后用类似数学表达式的形式直接输入系统的传递函数模型，下面将通过例子演示这两种输入方式。

例 2-17 考虑传递函数模型 $G(s) = \dfrac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24}$，用下面的语句就可以轻易地将该数学模型输入到 MATLAB 的工作空间。

```
>> num=[1 7 24 24]; den=[1 10 35 50 24]; % 分子多项式和分母多项式
   G=tf(num,den)    % 这样就能获得系统的数学模型 G
```

如果采用后一种输入方法，同样可以输入系统的传递函数模型，二者完全一致。

```
>> s=tf('s');  % 先定义 Laplace 算子 s，然后输入传递函数
   G=(s^3+7*s^2+24*s+24)/(s^4+10*s^3+35*s^2+50*s+24);
```

上面模型很容易输入，方法很直观，但如果分子或分母多项式给出的不是完全展开的形式，而是若干个因式的乘积，甚至包括其他运算，则采用前一种输入方式很烦琐，直接采用后一种方式显得更直观。下面通过两个例子可以演示这样的输入方法。

例 2-18 试将 $G(s)=\dfrac{7(s^2+5)}{(s+1)^3(s^2+2s+3)(s^2+7)}$ 模型输入到 MATLAB 工作空间。

求解 由于这里给出的是因式型的分子、分母，所以需要首先定义 s 算子，然后可以由下面语句直接输入该传递函数。

```
>> s=tf('s'); G=7*(s^2+5)/(s+1)^3/(s^2+2*s+3)/(s^2+7)
        % 或 G=7*(s^2+5)/((s+1)^3*(s^2+2*s+3)*(s^2+7))
```

可以得出系统的传递函数为

$$G(s)=\frac{7s^2+35}{s^7+5s^6+19s^5+51s^4+95s^3+115s^2+77s+21}$$

例 2-19 试将传递函数 $G(s)=\dfrac{s^3+5(s+5)^2+4}{s^3(s+1)[(s+2)^2+2]}$ 输入到 MATLAB 工作空间。

求解 可以看出，这里给出的传递函数分母含有多项式混合运算，因为多项式内部含有 $(s+5)^2+5$ 项，用第一种输入方式会很烦琐，所以可以用第二种方式直接利用算子法输入系统的传递函数模型

```
>> s=tf('s'); G=(s^3+5*(s+5)^2+4)/(s^3*(s+1)*((s+2)^2+2))
```

可以得出系统的传递函数为 $G(s)=\dfrac{s^3+5s^2+50s+129}{s^6+5s^5+10s^4+6s^3}$。

这里，传递函数对象 G 除了分子和分母多项式外，还有其他各种各样的属性，如 `Ts` 为采样周期属性，`ioDelay` 属性则表示其延迟时间常数。MATLAB 的 `tf` 对象还允许携带其他信息 (或属性)，其全部属性可以由 `set(tf)` 命令列出。

例 2-20 若系统的时间延迟常数为 $\tau=2$，即系统模型为 $G(s)\mathrm{e}^{-2s}$，则可以用下面的命令输入 `G.ioDelay=2`。

由前面的例子可以看出，在 MATLAB 语言环境中表示一个传递函数模型是很容易的。如果有了传递函数模型 G，还可以由`tfdata()`函数来提取系统的分子和分母多项式，即

```
>> [num,den]=tfdata(G,'v')  % 其中 'v' 表示想获得数值
```

该语句可以提取出分子和分母多项式系数向量

$\boldsymbol{n}=[0,0,0,1,5,50,129]$, $\boldsymbol{d}=[1,5,10,6,0,0,0]$

更简单地，还可以通过下面语句提取传递函数的分子和分母多项式。

```
>> num=G.num{1}; den=G.den{1}; % 可以直接提取分子和分母多项式
```

这里 `{1}` 实际上为 `{1,1}`，表示第 1 输入和第 1 输出之间的传递函数，该方法直接适合于多变量系统的描述。

3. 系统的零极点模型及 MATLAB 表示

零极点模型实际上是传递函数模型的另一种表现形式，对原系统传递函数的分子和分母分别进行分解因式处理，则可以得出系统的零极点模型为

$$G(s)=K\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)} \tag{2-2-13}$$

其中 K 称为系统的增益，$z_i\ (i=1,\cdots,m)$ 和 $p_i\ (i=1,\cdots,n)$ 分别称为系统的零点和极点。很显然，对实系数的传递函数模型来说，系统的零极点或者为实数，或者以共轭复数的形式出现。

在 MATLAB 下表示零极点模型的方法很简单，先用向量的形式输入系统的零点和极点，然后调用 zpk() 函数就可以输入这个零极点模型了。

$\boldsymbol{z}$=[z_1; z_2; $\cdots$; z_m]; $\boldsymbol{p}$=[p_1; p_2; $\cdots$; p_n];
G=zpk($\boldsymbol{z}$, $\boldsymbol{p}$, K);

其中前面两个语句分别输入系统的零点列向量 $\boldsymbol{z}$ 和极点列向量 $\boldsymbol{p}$，后面的语句可以由这些信息和系统增益构造出系统的零极点模型对象 G。零极点模型也有自己的诸多属性，包括采样周期属性 Ts 和延迟时间常数 ioDelay。

例 2-21 试输入零极点模型 $G(s)=\dfrac{6(s+5)(s+2+\mathrm{j}2)(s+2-\mathrm{j}2)}{(s+4)(s+3)(s+2)(s+1)}$。

求解 可以通过下面的 MATLAB 语句输入这个系统模型

```
>> P=[-1;-2;-3;-4]; % 注意应使用列向量，另外注意符号
   Z=[-5; -2+2i; -2-2i]; G=zpk(Z,P,6)
```

可以输入系统的零极点模型，并显示为 $G(s)=\dfrac{6(s+5)(s^2+4s+8)}{(s+1)(s+2)(s+3)(s+4)}$。

注意在 MATLAB 的零极点模型显示中，如果有复数零极点存在，则用二阶多项式来表示两个因式，而不直接展成一阶复数因式。

用 s=zpk('s') 定义零极点形式的 Laplace 算子，同样能输入零极点模型。

```
>> s=zpk('s');
   G=6*(s+5)*(s+2+2i)*(s+2-2i)/((s+1)*(s+2)*(s+3)*(s+4))
```

4. 多变量系统的传递函数矩阵模型

多变量系统的传递函数矩阵可以写成

$$\boldsymbol{G}(s)=\begin{bmatrix} g_{11}(s) & g_{12}(s) & \cdots & g_{1p}(s) \\ g_{21}(s) & g_{22}(s) & \cdots & g_{2p}(s) \\ \vdots & \vdots & \ddots & \vdots \\ g_{q1}(s) & g_{q2}(s) & \cdots & g_{qp}(s) \end{bmatrix} \tag{2-2-14}$$

其中 $g_{ij}(s)$ 可以定义为第 i 路输出信号对第 j 路输入信号的放大倍数，称为 (i,j) 子传递函数。多变量系统的传递函数矩阵的输入方法也很简单、直观，可以先输入各个子传递函数，然后用矩阵输入的命令就可以构造出系统的传递函数矩阵。

例 2-22 试将下面带有时间延迟的多变量传递函数矩阵[1]输入到 MATLAB 环境。

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

求解 对这样的多变量系统，只需先输入各个子传递函数矩阵，再按照常规矩阵的方式输入整个传递函数矩阵。具体的 MATLAB 命令如下:

```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
   g12=tf(0.924,[2.07 1]);
   g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
   g22=tf(-0.318,[2.93 1],'ioDelay',1.29);
   G=[g11, g12; g21, g22]; % 和矩阵定义一样，可以输入传递函数矩阵
```

这样的传递函数矩阵还可以由下面的方法输入，即输入各个不带延迟的子传递函数，构造传递函数矩阵，再重新赋值其 ioDelay 属性，亦即

```
>> g11=tf(0.1134,[1.78 4.48 1]); g12=tf(0.924,[2.07 1]);
   g21=tf(0.3378,[0.361 1.09 1]); g22=tf(-0.318,[2.93 1]);
   G=[g11, g12; g21, g22];
   G.ioDelay=[0.72 0; 0.3, 1.29]; % 再设置时间延迟矩阵
```

其中的 (2,1) 子传递函数可以用 `G(2,1)` 语句直接提取出来。

5. 传递函数和零极点模型的相互转换

显而易见，式 (2-2-12) 中定义的传递函数模型和式 (2-2-13) 中表示的零极点模型是描述同样微分方程模型的不同表示形式。对零极点模型进行展开处理则得出传递函数模型，而对传递函数进行因式化则可以得出系统的零极点模型。MATLAB 中提供了更简洁的方法。假设 G 和 G_1 分别表示传递函数和零极点模型，则它们之间的相互转换可以由 G_1=`zpk(`G`)` 和 G=`tf(`G_1`)` 直接求出。

6. 二阶典型系统的阶跃响应解析解推导

在控制理论教材中经常将二阶典型系统 $G(s)=\dfrac{\omega_{\rm n}^2}{s^2+2\zeta\omega_{\rm n}s+\omega_{\rm n}^2}$ 作为研究对象，研究其阶跃响应曲线。根据不同的 ζ 取值，将系统分为无阻尼、欠阻尼、等阻尼和过阻尼四种情况考虑，每种情况有自己的阶跃响应解析表达式。

利用 Laplace 变换与反变换技术，给出下面的语句

```
>> syms z s; syms wn positive; G=wn^2/(s^2+2*z*wn*s+wn^2);
```

```
R=laplace(sym(1)); y=ilaplace(G*R)
```

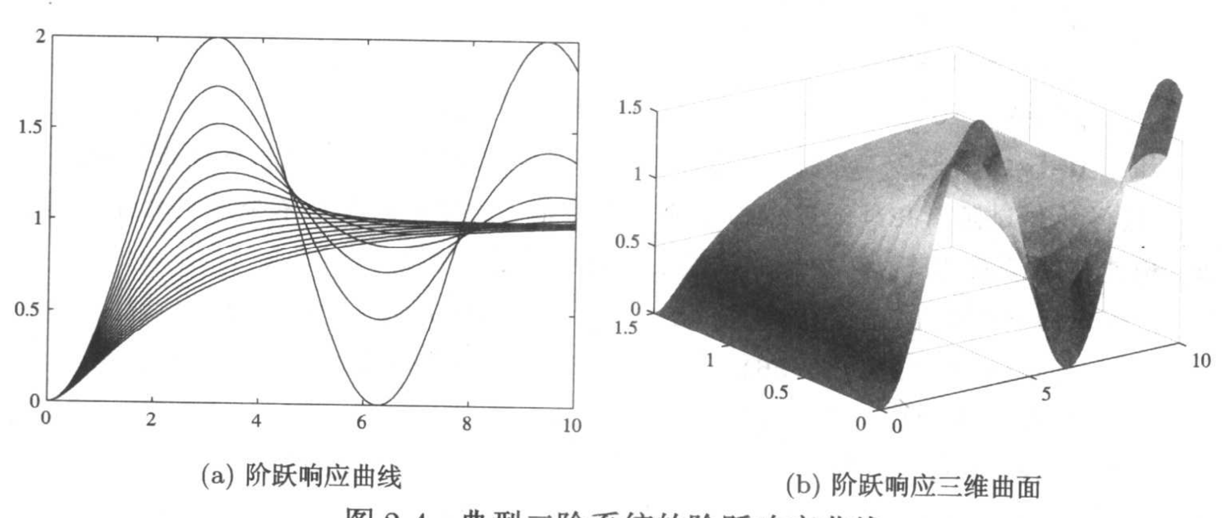
则可以直接推导出二阶系统阶跃响应的统一表达式为

$$y(t) = 1 - \omega_{\rm n}{\rm e}^{-\zeta\omega_{\rm n}t}\left[\frac{\cosh(\omega_{\rm d}t)}{\omega_{\rm n}} + \frac{\zeta\sinh(\omega_{\rm d}t)}{\omega_{\rm d}}\right] \tag{2-2-15}$$

式中 $\omega_{\rm d} = \sqrt{\zeta^2-1}\,\omega_{\rm n}$。注意，该通解不能处理 $\zeta=1$ 时的系统响应，但在数值上用 $\zeta = 1-\epsilon$ 即可以求解，或采用式 $y(t) = 1 + (-\omega_{\rm n}t-1){\rm e}^{-\omega_{\rm n}t}$ 直接求解。

例 2-23 试绘制在 $\omega_{\rm n}=1$ 时，不同 ζ 值下的对象二阶系统阶跃响应表示。

求解 可以令 $\omega_{\rm n}=1$，这时用循环结构就可以得出响应数据，并绘制出阶跃响应曲线，如图 2-4 (a) 所示。另外，利用 MATLAB 语言的三维表面图绘制技术还可以得出阶跃响应是三维曲面表示，如图 2-4 (b) 所示。



(a) 阶跃响应曲线　　(b) 阶跃响应三维曲面

图 2-4　典型二阶系统的阶跃响应曲线

```
>> wn=1; zeta=[0:.1:0.9 1-eps, 1.1:0.1:1.5]; t=0:0.05:10; y=[];
   for z=zeta, wd=sqrt(z^2-1)*wn;
      y=[y; 1-wn*exp(-z*wn*t).*[cosh(wd*t)/wn+z*sinh(wd*t)/wd]];
   end
   plot(t,y), figure, surf(t,zeta,y), shading flat
```

2.3 Fourier 变换及反变换

类似于前面介绍的 Laplace 变换，可以将时域信号通过积分变换的方法变换成复变量 s 的函数，Fourier 变换引入频域变量 ω，将时域函数映射成频域变量 ω 的函数。Fourier 变换的性质类似于 Laplace 变换，基于 MATLAB 的求解方法也类似于 Laplace 变换。本节将介绍 Fourier 变换的定义、性质及求解方法，并将介

绍 MATLAB 不直接支持的，而在实际中有用的各种其他形式的 Fourier 变换与求解方法。

2.3.1 给定函数的 Fourier 级数展开

给定周期性数学函数 $f(x)$，其中，$x \in [-L, L]$，且周期为 $T = 2L$，可以人为地对该函数在其他区间上进行周期延拓，使得 $f(x) = f(kT + x)$，k 为任意整数，这样可以根据需要将其写成下面的级数形式。

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi}{L} x + b_n \sin \frac{n\pi}{L} x \right) \tag{2-3-1}$$

其中，

$$\begin{cases} a_n = \dfrac{1}{L} \displaystyle\int_{-L}^{L} f(x) \cos \dfrac{n\pi x}{L} \mathrm{d}x, & n = 0, 1, 2, \cdots \\ b_n = \dfrac{1}{L} \displaystyle\int_{-L}^{L} f(x) \sin \dfrac{n\pi x}{L} \mathrm{d}x, & n = 1, 2, 3, \cdots \end{cases} \tag{2-3-2}$$

该级数称为 Fourier 级数，而 a_n, b_n 又称为 Fourier 系数。若 $x \in (a, b)$，则可以计算出 $L = (b - a)/2$，引入新变量 $\hat{x}$，使得 $x = \hat{x} + L + a$，则可以将 $f(\hat{x})$ 映射成 $(-L, L)$ 区间上的函数，可以对之进行 Fourier 级数展开，再将 $\hat{x} = x - L - a$ 转换成 x 的函数即可。

MATLAB 和 Maple 语言均未直接提供求解 Fourier 系数与级数的现成函数。其实由上述公式不难编写出解析或数值的 Fourier 级数求解函数。其中解析函数如下：

```
function [A,B,F]=fseries(f,x,p,a,b)
if nargin==3, a=-pi; b=pi; end
L=(b-a)/2; if a+b, f=subs(f,x,x+L+a); end
A=int(f,x,-L,L)/L; B=[]; F=A/2;
for n=1:p
   an=int(f*cos(n*pi*x/L),x,-L,L)/L;
   bn=int(f*sin(n*pi*x/L),x,-L,L)/L; A=[A, an]; B=[B,bn];
   F=F+an*cos(n*pi*x/L)+bn*sin(n*pi*x/L);
end
if a+b, F=subs(F,x,x-L-a); end
```

该函数的调用格式为

```
[A,B,F]=fseries(f,x,p,a,b)
```

其中，f 为给定函数，x 为自变量，p 为展开项数，a, b 为 x 的区间，可以省略取其默认值 $[-\pi, \pi]$，得出的 $\boldsymbol{A}$, $\boldsymbol{B}$ 为 Fourier 系数，F 为展开式。仿照解析解 `fseries()` 函数其实不难写出其数值版，有兴趣的读者可以自己试一试。

例 2-24 试求给定函数 $y = x(x-\pi)(x-2\pi), x \in (0, 2\pi)$ 的 Fourier 级数展开。

求解 上述给定函数的 Fourier 级数展开可以很自然地用下面的语句得出。

```
>> syms x; f=x*(x-pi)*(x-2*pi);
   [A,B,F]=fseries(f,x,12,0,2*pi); latex(F)
```

这样，可以得出前 12 项的 Fourier 级数展开为

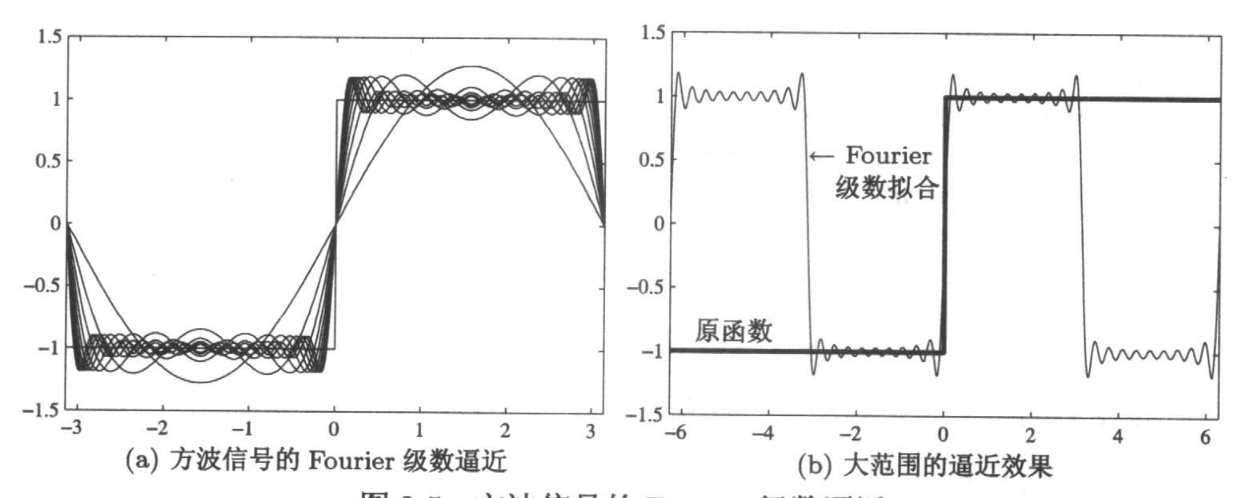
$$f(x) = 12\sin x + \frac{3}{2}\sin 2x + \frac{4}{9}\sin 3x + \frac{3}{16}\sin 4x + \frac{12}{125}\sin 5x + \frac{1}{18}\sin 6x + \frac{12}{343}\sin 7x + \frac{3}{128}\sin 8x + \frac{4}{243}\sin 9x + \frac{3}{250}\sin 10x + \frac{12}{1331}\sin 11x + \frac{1}{144}\sin 12x$$

其实，该展开的解析表达式为 $f(x) = \sum\limits_{n=1}^{\infty} \dfrac{12}{n^3}\sin nx$。

例 2-25 考虑 $(-\pi, \pi)$ 区间的方波信号，假设 $x \geqslant 0$ 时 $y = 1$，否则 $y = -1$，试对该方波信号进行 Fourier 级数拟合，并观察用多少项能有较好的拟合效果。

求解 给定的函数可以由 $f(x) = |x|/x$ 表示，故由下面语句可以容易地生成 x-轴数据点，求出理论的方波数值，并通过不同阶次的 Fourier 级数展开去拟合原来的方波函数。得出的曲线如图 2-5 (a) 所示。

```
>> syms x; f=abs(x)/x;  % 定义方波信号
   xx=[-pi:pi/200:pi]; xx=xx(xx~=0); xx=sort([xx,-eps,eps]);
   yy=subs(f,x,xx); plot(xx,yy), hold on
   for n=1:20
      [a,b,f1]=fseries(f,x,n); y1=subs(f1,x,xx); plot(xx,y1)
   end
```



(a) 方波信号的 Fourier 级数逼近　　(b) 大范围的逼近效果

图 2-5　方波信号的 Fourier 级数逼近

从得出的结果看，当阶次等于 10 左右就能得出较好的拟合，再增加阶次也不会有显著的改善效果。取 $n = 14$，则 Taylor 级数展开可以由下面的语句具体得出。

```
>> [a,b,f1]=fseries(f,x,14); f1
```

Taylor 幂级数展开可以具体表示成

$$f(x) \approx 4\frac{\sin x}{\pi}+\frac{4\sin 3x}{3\pi}+\frac{4\sin 5x}{5\pi}+\frac{4\sin 7x}{7\pi}+\frac{4\sin 9x}{9\pi}+\frac{4\sin 11x}{11\pi}+\frac{4\sin 13x}{13\pi}$$

从该结果可以总结出一般的展开公式为 $f(x)=\dfrac{4}{\pi}\sum\limits_{k=1}^{\infty}\dfrac{\sin(2k-1)x}{2k-1}$。另外，应该注意这里采用的 Fourier 级数的拟合范围。因为 Fourier 级数是假定 $f(t)$ 是周期函数，而原函数本身则不是，故在指定的 $(-\pi,\pi)$ 范围内拟合效果比较满意，但此范围之外则完全不同，见图 2-5 (b)。

```
>> xx=[-2*pi:pi/200:2*pi]; xx=xx(xx~=0); xx=sort([xx,-eps,eps]);
   yy=subs(f,x,xx); y1=subs(f1,x,xx); plot(xx,y1,xx,yy)
```

2.3.2 Fourier 变换及反变换定义与性质

Fourier 变换的一般定义为

$$\mathscr{F}[f(x)]=\int_{-\infty}^{\infty}f(x)\mathrm{e}^{-\mathrm{j}\omega x}\mathrm{d}x=F_1(\omega) \tag{2-3-3}$$

如果已知 Fourier 变换式子 $F(\omega)$，则可以由 Fourier 反变换公式反演出 $f(t)$ 函数为

$$f(x)=\mathscr{F}^{-1}[F_1(\omega)]=\frac{1}{2\pi}\int_{-\infty}^{\infty}F_1(\omega)\mathrm{e}^{\mathrm{j}\omega x}\mathrm{d}\omega \tag{2-3-4}$$

这里仍将不加证明地列出一些 Fourier 变换的性质：

① **线性性质** 若 a 与 b 均为标量，则 $\mathscr{F}[af(t)\pm bg(t)]=a\mathscr{F}[f(t)]\pm b\mathscr{F}[g(t)]$。

② **平移性质** $\mathscr{F}[f(t\pm a)]=\mathrm{e}^{\pm\mathrm{j}a\omega}F(\omega)$。

③ **复域平移性质** $\mathscr{F}[\mathrm{e}^{\pm\mathrm{j}at}f(t)]=F(\omega\mp a)$。

④ **微分性质** $\mathscr{F}[\mathrm{d}f(t)/\mathrm{d}t]=\mathrm{j}\omega F(\omega)$，一般地，$n$ 阶微分可以由下式求出。

$$\mathscr{F}\left[\frac{\mathrm{d}^n}{\mathrm{d}t^n}f(t)\right]=(\mathrm{j}\omega)^n\mathscr{F}[f(t)] \tag{2-3-5}$$

⑤ **积分性质** $\mathscr{F}\left[\int_{-\infty}^{t}f(\tau)\mathrm{d}\tau\right]=\dfrac{F(\omega)}{\mathrm{j}\omega}$，一般地，函数 $f(t)$ 的 n 重积分的 Fourier 变换可以由下式求出。

$$\mathscr{F}\left[\int_{-\infty}^{t}\cdots\int_{-\infty}^{t}f(\tau)(\mathrm{d}\tau)^n\right]=\frac{\mathscr{F}[f(t)]}{(\mathrm{j}\omega)^n} \tag{2-3-6}$$

⑥ **尺度变换** $\mathscr{F}[f(at)]=\dfrac{1}{a}F\left(\dfrac{\omega}{a}\right)$。

⑦ **卷积性质** $\mathscr{F}[f(t)*g(t)]=\mathscr{F}[f(t)]\mathscr{F}[g(t)]$，卷积定义仍为式 (2-2-5)。

2.3.3　Fourier 变换的计算机求解

和 Laplace 变换一样，应该先申明符号变量，并定义出原函数为 Fun，这样就可以按如下的格式调用 Fourier 变换求解函数 fourier()，得出该函数的 Fourier 变换式。

```
F=fourier(Fun)            % 按默认变量进行 Fourier 变换
F=fourier(Fun, v, u)      % 将 v 的函数变换成 u 的函数
```

给出了 Fourier 变换表达式，则可以通过 ifourier() 函数求解该函数的 Fourier 反变换问题。该函数的具体调用格式为

```
f=ifourier(Fun)           % 按默认变量进行 Fourier 反变换
f=ifourier(Fun, u, v)     % 将 u 的函数变换成 v 的函数
```

从上面的语句可以看出，如果定义了已知函数，则可以用一个语句求解出其 Fourier 变换或反变换式子，变换函数的调用和 Laplace 变换一样简单。

例 2-26 考虑 $f(t)=1/(t^2+a^2)$, $a>0$，试写出该函数的 Fourier 变换式。

求解 可以用下面的语句得出原函数的 Fourier 变换。

```
>> syms t w; syms a positive; f=1/(t^2+a^2); F=fourier(f,t,w)
```

这样得出的结果为 $\dfrac{\pi}{a}\left[\mathrm{e}^{-a\omega}\mathrm{Heaviside}(\omega)+\mathrm{e}^{a\omega}\mathrm{Heaviside}(-\omega)\right]$，其中 Heaviside($\omega$) 函数为 ω 的阶跃函数，又称为 Heaviside 函数，其数学定义为

$$\mathrm{Heaviside}(\omega)=\begin{cases}1 & \omega\geqslant 0\\ 0 & \text{其他 }\omega\text{ 取值}\end{cases}\tag{2-3-7}$$

假设 $\omega>0$，则 F 可以简化成 $\pi\mathrm{e}^{-a\omega}/a$，若 $\omega<0$，则 F 可以简化成 $\pi\mathrm{e}^{a\omega}/a$，故可以将上述结果写成 $\mathscr{F}[f(t)]=\dfrac{\pi\mathrm{e}^{-a|\omega|}}{a}$。

然而，这样的最简表达式是通过 MATLAB 语言的自动化简功能无法得出的。对得出的结果进行 Fourier 反变换，则将还原出原函数。

```
>> syms t w; syms a positive; f=pi*exp(-a*abs(w))/a; ifourier(f)
```

例 2-27 假设时域函数为 $f(t)=\sin^2(at)/t$, $a>0$，试求出其 Fourier 变换。

求解 由给定的式子，可以用下面的语句获得原函数的 Fourier 变换。

```
>> syms t w; syms a positive; f=sin(a*t)^2/t; fourier(f,t,w)
```

这样，得出的结果为 $\dfrac{\mathrm{j}\pi}{2}\Big[\mathrm{Heaviside}(\omega-2a)+\mathrm{Heaviside}(\omega+2a)-2\mathrm{Heaviside}(\omega)\Big]$。可见，该结果仍然依赖于 Heaviside() 函数，所以理解 Heaviside() 函数将使得用户能得到更简单的形式。所以由上面的结果可知：当 $\omega>2a$，则 3 个 Heaviside() 函数的值均为 1，故 $F(\omega)=0$。若 $\omega\leqslant-2a$，则 3 个函数的值均为 0，故 $F(\omega)=0$。若 $0<\omega<2a$，则第 2 和第 3 个 Heaviside() 的值为 1，故 $F(\omega)=-\mathrm{j}\pi/2$。当 $0>\omega>-2a$，则 $F(\omega)=\mathrm{j}\pi/2$。综上所述，原函数的 Fourier 变换可以化简成

$$\mathscr{F}[f(t)]=\begin{cases}0, & |\omega|>2a\\ -\mathrm{j}\pi\mathrm{sgn}(\omega)/2, & |\omega|<2a\end{cases}$$

例 2-28 试求出分段函数 $f(t)=\begin{cases}\mathrm{e}^{-t}\sin 2t, & t\geqslant 0\\ 0, & t<0\end{cases}$ 的 Fourier 变换。

求解 该已知函数可以由单一的符号函数 $f(t)=\mathrm{e}^{-t}\sin 2t\mathrm{Heaviside}(t)$ 表示，故其 Fourier 变换可以由下面的语句直接求出

```
>> syms t; f=exp(-t)*sin(2*t)*heaviside(t); simple(fourier(f))
```

其变换结果为 $\dfrac{-2}{-5+\omega^2-2\mathrm{j}\omega}$。

例 2-29 试求出有限区间函数 $f(t)=\begin{cases}1-|t|, & |t|\leqslant 1\\ 0, & |t|>1\end{cases}$ 的 Fourier 变换。

求解 由给出的表达式可见，很难用单一的 MATLAB 符号函数描述整个 $f(t)$ 函数，从而调用 fourier() 函数求解原问题，所以应该对原始思路做一个调整，不采用该函数，而直接由定义求解原问题，这样可以给出下面的语句

```
>> syms t w; F=simple(int((1-abs(t))*exp(-sqrt(-1)*w*t),t,-1,1))
```

可以得出 $F(\omega)=\mathscr{F}[f(t)]=2(1-\cos\omega)/\omega^2$。

2.3.4 Fourier 正弦和余弦变换

Fourier 正弦变换的一般定义为

$$\mathscr{F}_{\sin}[f(t)]=\int_0^{\infty}f(t)\sin(\omega t)\mathrm{d}t=F_{\mathrm{s}}(\omega)\tag{2-3-8}$$

Fourier 余弦变换的一般定义为

$$\mathscr{F}_{\cos}[f(t)]=\int_0^{\infty}f(t)\cos(\omega t)\mathrm{d}t=F_{\mathrm{c}}(\omega)\tag{2-3-9}$$

相应地，还可以如下定义 Fourier 正弦和余弦反变换为

$$\mathscr{F}_{\sin}^{-1}[F_{\mathrm{s}}(t)]=\frac{2}{\pi}\int_{-\infty}^{\infty}F_{\mathrm{s}}(\omega)\sin(\omega t)\mathrm{d}\omega\tag{2-3-10}$$

$$\mathscr{F}_{\cos}^{-1}[F_c(t)] = \frac{2}{\pi}\int_{-\infty}^{\infty} F_c(\omega)\cos(\omega t)\mathrm{d}\omega \tag{2-3-11}$$

类似于 Fourier 变换，还可以定义出对称的 Fourier 正弦和余弦变换，即在正变换中乘以 $\sqrt{2/\pi}$，反变换除以 $\sqrt{2/\pi}$，所以若能得到前面定义的变换，就可以转换成对称定义的变换。

MATLAB 语言的符号运算工具箱中并未直接提供余弦 Fourier 变换的函数，所以可以考虑采用符号积分的方法直接求取余弦 Fourier 变换。下面将提供具体例子演示正弦和余弦 Fourier 变换的推导方法。

例 2-30 试求出 $f(t)=t^n\mathrm{e}^{-at}, a>0, n=1,2,\cdots,8$ 的余弦 Fourier 变换。

求解 解决这样的问题可以采用循环结构，对不同的 i 值，可以用直接积分的算法求取 Fourier 余弦变换，并将得出的结果进行化简，如表 2-1 所示。

```
>> syms t w; syms a positive
   for i=1:8
      f=t^i*exp(-a*t); F=int(f*cos(w*t),t,0,inf); latex(simple(F))
   end
```

在前面给出的命令中使用了积分解析求解的 int() 函数，该函数的调用格式将在第 4 章中详细介绍。其实，按照数学手册[2]中给出的公式，对整数 n，可以得出

$$\mathscr{F}_{\cos}\left[t^n\mathrm{e}^{-at}\right] = n!\left(\frac{a}{a^2+\omega^2}\right)^{n+1}\sum_{m=0}^{[n/2]}(-1)^m C_{n+1}^{2m+1}\left(\frac{\omega}{a}\right)^{2m+1} \tag{2-3-12}$$

表 2-1 n 取不同值时 Fourier 余弦变换结果

n 值	$\mathscr{F}_{\cos}[f(t)]$
1~4	$\dfrac{a^2-\omega^2}{(a^2+\omega^2)^2}$, $-\dfrac{2(-a^2+3\omega^2)a}{(a^2+\omega^2)^3}$, $\dfrac{6(-a^2+2a\omega+\omega^2)(-a^2-2a\omega+\omega^2)}{(a^2+\omega^2)^4}$, $\dfrac{24a(a^4-10a^2\omega^2+5\omega^4)}{(a^2+\omega^2)^5}$
5,6	$-\dfrac{120(-a+\omega)(a+\omega)(a^2-4\omega a+\omega^2)(a^2+4\omega a+\omega^2)}{(a^2+\omega^2)^6}$, $-\dfrac{720(-a^6+21a^4\omega^2-35\omega^4a^2+7\omega^6)a}{(a^2+\omega^2)^7}$
7	$\dfrac{5040\left(a^4+4a^3\omega-6a^2\omega^2-4a\omega^3+\omega^4\right)\left(a^4-4a^3\omega-6a^2\omega^2+4a\omega^3+\omega^4\right)}{(a^2+\omega^2)^8}$
8	$\dfrac{40320a\left(-a^2+3\omega^2\right)\left(-a^6+33a^4\omega^2-27a^2\omega^4+3\omega^6\right)}{(a^2+\omega^2)^9}$

MATLAB 语言并未直接提供 Fourier 正弦、余弦变换的现成函数，而 Maple 语言提供了 Fourier 正弦、余弦变换的函数，故可以用 MATLAB 的符号运算工

具箱直接调用 Maple 中的现成函数。具体格式为

```
F=maple('fouriersin',f,t,ω)        % 求解 Fourier 正弦变换
F=maple('fouriercos',f,t,ω)        % 求解 Fourier 余弦变换
f=maple('invfouriersin',F,ω,t)     % 求解 Fourier 反正弦变换
f=maple('invfouriercos',F,ω,t)     % 求解 Fourier 反余弦变换
```

其中，maple() 函数是 MATLAB 符号运算工具箱中的函数，允许用户直接调用 Maple 中的有关函数，'fouriersin' 等是 Maple 中的函数名，其余参数是 Maple 中参数的格式。

例 2-31 重新考虑例 2-30 中给出的分段函数，令 $n = 6$，采用 Maple 中提供的现成函数求解 Fourier 余弦变换问题，并试着对结果进行 Fourier 余弦反变换。

求解 利用 Maple 中的 Fourier 余弦变换函数和反变换函数可以得出如下的结果:

```
>> syms t w; syms a positive
   f=t^6*exp(-a*t); F=maple('fouriercos',f,t,w); latex(F)
```

该结果可以直接表示为 $\mathscr{F}_{\cos}[f(t)] = 720\sqrt{\dfrac{2}{\pi}}\dfrac{a^7}{(a^2+\omega^2)^7}\left(1-21\dfrac{\omega^2}{a^2}+35\dfrac{\omega^4}{a^4}-7\dfrac{\omega^6}{a^6}\right)$。

和表 2-1 的结果相比可见，由这种方法得出的不是式 (2-3-9) 中给出的变换，而是对称处理的变换结果，亦即将该式得出的结果乘以 $\sqrt{2/\pi}$ 后得出的结果。

例 2-32 试求取分段函数 $f(t) = \begin{cases} \cos(t), & 0 < x < a \\ 0, & \text{其他} \end{cases}$ 的 Fourier 余弦变换。

求解 用 Maple 中变换方法的难点在于表达分段函数，所以仍可以考虑直接通过定义进行变换的方法。研究定义式子可以立即发现，式 (2-3-9) 的被积函数在 $t \in (a, \infty)$ 区间的值为 0，这样，其积分亦为 0，故整个积分问题就变成 $t \in (0, a)$ 区间的积分问题了，故可以用下面的语句求出该函数的 Fourier 余弦变换为

```
>> syms t w; syms a positive
   f=cos(t); F=simple(int(f*cos(w*t),t,0,a)); latex(F)
```

得出的结果为 $\mathscr{F}_{\cos}[f(t)] = \dfrac{(1+\omega)\sin(-a+a\omega)+(-1+\omega)\sin(a+a\omega)}{-2+2\omega^2}$，该式还可以进一步手工化简为 $\mathscr{F}_{\cos}[f(t)] = \dfrac{\sin(-a+a\omega)}{2(1+\omega)}+\dfrac{\sin(a+a\omega)}{2(1-\omega)}$。

2.3.5 离散 Fourier 正弦、余弦变换

离散 Fourier 正弦、余弦变换又称为有限 Fourier 正弦、余弦变换，和前面介绍的 Fourier 正弦、余弦变换相比，其积分区间从 $t \in (0, \infty)$ 变成了 $t \in (0, a)$，故

其定义为

$$F_{\mathrm{s}}(k)=\int_0^a f(t)\sin\frac{k\pi t}{a}\mathrm{d}t,\ \ F_{\mathrm{c}}(k)=\int_0^a f(t)\cos\frac{k\pi t}{a}\mathrm{d}t \tag{2-3-13}$$

相应地，可以定义出有限 Fourier 正弦、余弦反变换为

$$f(t)=\frac{2}{a}\sum_{k=1}^{\infty}F_{\mathrm{s}}(k)\sin\frac{k\pi t}{a} \tag{2-3-14}$$

$$f(t)=\frac{1}{a}F_{\mathrm{c}}(0)+\frac{2}{a}\sum_{k=1}^{\infty}F_{\mathrm{c}}(k)\cos\frac{k\pi t}{a} \tag{2-3-15}$$

和前面定义的不同，反变换不再是积分式子，而是无穷级数的求和。下面通过例子介绍如何用 MATLAB 及其符号运算工具箱求取给定函数的有限变换。

例 2-33 考虑分段函数 $f(t)=\begin{cases} t & t\leqslant a/2 \\ a-t & t>a/2 \end{cases}$，其中，$a>0$，试求其离散 Fourier 正弦变换。

求解 函数的离散 Fourier 正弦变换可以由下面的语句直接求出。

```
>> syms t k; syms a positive; f1=t; f2=a-t;
   Fs=int(f1*sin(k*pi*t/a),t,0,a/2)+int(f2*sin(k*pi*t/a),t,a/2,a);
   simple(Fs)
```

因为符号运算工具箱不直接支持整数变量，所以该方程只能最终化简成

$$\mathscr{F}_{\sin}[f(t)]=\frac{a^2(2\sin(1/2k\pi)-\sin(k\pi))}{k^2\pi^2}$$

考虑到 k 为整数，故有 $\sin(k\pi)\equiv 0$，该式可以手工化简为 $\mathscr{F}_{\sin}[f(t)]=\dfrac{2a^2}{k^2\pi^2}\sin\dfrac{k\pi}{2}$。

2.4 Z 变换及反变换

严格说来，Z 变换并不属于积分变换，但由于其定义、性质和求解方法也类似于 Laplace 方法，并且该方法可以在描述离散系统数学模型中起着重要的作用，所以本节将介绍 Z 变换，并引入离散传递函数表示方法，离散、连续传递函数的双线性变换方法。

2.4.1 Z 变换及反变换定义与性质

离散序列信号 $f(k)$ 的 Z 变换可以定义为

$$\mathscr{Z}[f(k)]=\sum_{i=0}^{\infty}f(k)z^{-k}=F(z) \tag{2-4-1}$$

类似于前面介绍的 Laplace 变换和 Fourier 变换，Z 变换也有很多性质，仍不加证明地列出一些性质如下：

① **线性性质** 若 a 与 b 为标量，则 $\mathscr{Z}[af(k)\pm bg(k)] = a\mathscr{Z}[f(k)]\pm b\mathscr{Z}[g(k)]$。

② **时域平移性质** $\mathscr{Z}[f(k-n)] = z^{-n}F(z)$。

③ s **域比例性质** $\mathscr{Z}[r^{-k}f(k)] = F(rz)$。

④ **频域微分性质** $\mathscr{Z}[kf(k)] = -z\mathrm{d}F(z)/\mathrm{d}z$。

⑤ **频域积分性质** $\mathscr{Z}[f(k)/k] = \displaystyle\int_z^\infty F(\omega)/\omega\mathrm{d}\omega$。

⑥ **初值性质** $\lim\limits_{k\to 0} f(k) = \lim\limits_{z\to\infty} F(z)$。

⑦ **终值性质** 如果 $F(z)$ 无单位圆外的极点，则 $\lim\limits_{k\to\infty} f(k) = \lim\limits_{k\to 1}(z-1)F(z)$。

⑧ **卷积性质** $\mathscr{Z}[f(k)*g(k)]=\mathscr{Z}[f(k)]\mathscr{Z}[g(k)]$，离散卷积算子 $*$ 定义为

$$f(k) * g(k) = \sum_{l=0}^{\infty} f(k)g(k-l) \tag{2-4-2}$$

给定 Z 变换式子 $F(z)$，则其 Z 反变换的数学表示为

$$f(k) = \mathscr{Z}^{-1}[f(k)] = \frac{1}{2\pi \mathrm{j}}\oint F(z)z^{k-1}\mathrm{d}z \tag{2-4-3}$$

2.4.2 Z 变换的计算机求解

利用 MATLAB 的符号运算工具箱，则 Laplace 变换、Z 变换及反变换可以很容易地求取出来，掌握这样的工具可以免除复杂问题的手工推导，既节省时间又能避免底层的低级错误。利用符号运算工具箱中提供的 `ztrans()` 和 `iztrans()` 函数可用得出给定函数的 Z 变换及反变换。这两个函数的调用格式为

```
F=ztrans(Fun, k, z)    % 按默认变量进行 Z 变换
F=iztrans(Fun, z, k)   % Z 反变换，将 z 的函数变换成 k 的函数
```

若原函数只有一个变量，则调用时无需给出 k 和 z。

例 2-34 求解 $f(kT) = akT-2+(akT+2)\mathrm{e}^{-akT}$ 函数的 Z 变换问题。

求解 原函数的 Z 变换可以用下面的语句来完成。

```
>> syms a T k  % 声明符号变量
   f=a*k*T-2+(a*k*T+2)*exp(-a*k*T); F=ztrans(f) % 定义并求解
```

该结果为 $\mathscr{Z}[f(kT)] = \dfrac{aTz}{(z-1)^2} - 2\dfrac{z}{z-1} + \dfrac{aTz\mathrm{e}^{-aT}}{(z-\mathrm{e}^{-aT})^2} + 2z\mathrm{e}^{aT}\left(\dfrac{z}{\mathrm{e}^{-aT}}-1\right)^{-1}$。

例 2-35 假设零阶保持器的 Laplace 变换表示[3]为 $G_{\rm h}(s)=\dfrac{1-{\rm e}^{-Ts}}{s}$，并已知传递函数 $G(s)=\dfrac{K}{s(s+a)}$，试导出受控对象的 Z 变换表达式 $\mathscr{Z}[G_{\rm h}(s)G(s)]$。

求解 由定义知 $z={\rm e}^{Ts}$，故要求的 Z 变换可以改写成 $(1-z^{-1})\mathscr{Z}[G(s)/s]$

```
>> syms T s z a K; G=K/s/(s+a); g=ilaplace(G/s);
   Gz=simple((1-z^(-1))*ztrans(g))
```

上面的语句可以直接导出所需的 Z 变换为

$$\mathscr{Z}[G_{\rm h}(s)G(s)]=\frac{(az-z+z{\rm e}^{-a}+1-a{\rm e}^{-a}-{\rm e}^{-a})\,K}{a^2\,(z^2-z{\rm e}^{-a}-z+{\rm e}^{-a})}$$

例 2-36 考虑 $F(z)=q/(z^{-1}-p)^m$ 函数的 Z 反变换问题，这里可以对不同的 m 值进行反变换，并总结出一般规律。

求解 根据要求，可以用符号运算工具箱求出 $m=1,2,\cdots,8$ 的 Z 反变换。

```
>> syms p q z
   for i=1:8, disp(simple(iztrans(q/(1/z-p)^i))), end
```

则可以得出下面一系列的结果

$-q/p(1/p)^n$

$q/p^2(1+n)(1/p)^n$

$-1/2q(1/p)^n(1+n)(2+n)/p^3$

$1/6q(1/p)^n(3+n)(2+n)(1+n)/p^4$

$-1/24q(1/p)^n(4+n)(3+n)(2+n)(1+n)/p^5$

$1/120q(1/p)^n(5+n)(4+n)(3+n)(2+n)(1+n)/p^6$

$-1/720q(1/p)^n(6+n)(5+n)(4+n)(3+n)(2+n)(1+n)/p^7$

$1/5040q(1/p)^n(7+n)(6+n)(5+n)(4+n)(3+n)(2+n)(1+n)/p^8$

总结上述结果的规律，可以写出一般的 Z 反变换结果为

$$\mathscr{Z}^{-1}\left[\frac{q}{(z^{-1}-p)^m}\right]=\frac{(-1)^m q}{(m-1)!\,p^{n+m}}\prod_{i=1}^{m-1}(n+i) \tag{2-4-4}$$

2.4.3 离散时间系统的建模

类似于连续系统，若对描述离散系统的差分方程进行 Z 变换，则可以推导出离散系统的传递函数模型，这里将演示差分方程到离散系统传递函数的转换，引入离散传递函数的概念及 MATLAB 表示方法，并介绍连续、离散系统的双线性变换方法。

1. 线性离散系统的差分方程模型

单变量线性离散系统可以由差分方程表示

$$\begin{aligned}a_1y(kT)+a_2y[(k-1)T]+\cdots+a_ny[(k-n-1)T]+a_{n+1}y[(k-n)T]= \\ b_0u(kT)+b_1u[(k-1)T]+\cdots+b_{n-1}u[(k-n-1)T]+b_nu[(k-n)T]\end{aligned} \tag{2-4-5}$$

式中 T 为离散系统的采样周期。

2. 离散系统的传递函数模型及 MATLAB 表示

类似于微分方程，差分方程的直接求解也是很困难的，所以可以借助于 Z 变换的功能，将离散系统的差分方程转换成 Z 变换形式。离散系统的离散传递函数模型可以定义为

$$H(z)=\frac{b_0z^n+b_1z^{n-1}+\cdots+b_{n-1}z+b_n}{a_1z^n+a_2z^{n-1}+\cdots+a_nz+a_{n+1}} \tag{2-4-6}$$

在 MATLAB 语言中，输入离散系统的传递函数模型和连续系统传递函数模型一样简单，只需分别按要求输入系统的分子和分母多项式，就可以利用 `tf()` 函数将其输入到 MATLAB 环境。和连续传递函数不同的是，同时还需要输入系统的采样周期 T，具体语句如下：

```
num=[b0,b1,···,bn-1,bn];  den=[a1,a2,···,an,an+1];
H=tf(num,den,'Ts',T);
```

其中 T 应该输入为实际的采样周期数值，H 为离散系统传递函数模型。此外，仿照连续系统传递函数的算子输入方法，定义算子 `z=tf('z',T)`，则可以用数学表达式形式输入系统的离散传递函数模型。

例 2-37 试输入离散系统的传递函数模型 $H(z)=\dfrac{6z^2-0.6z-0.12}{z^4-z^3+0.25z^2+0.25z-0.125}$，其中采样周期为 $T=0.1$ 秒。

求解 由下面的语句将其输入到 MATLAB 工作空间

```
>> num=[6 -0.6 -0.12]; den=[1 -1 0.25 0.25 -0.125];
   H=tf(num,den,'Ts',0.1)    % 输入并显示系统的传递函数模型
```

该模型还可以采用算子方式直接输入

```
>> z=tf('z',0.1);
   H=(6*z^2-0.6*z-0.12)/(z^4-z^3+0.25*z^2+0.25*z-0.125);
```

离散系统的时间延迟模型和连续系统不同，一般可以写成

$$H(z)=\frac{b_0z^n+b_1z^{n-1}+\cdots+b_{n-1}z+b_n}{a_1z^n+a_2z^{n-1}+\cdots+a_nz+a_{n+1}}z^{-m} \tag{2-4-7}$$

这就要求实际延迟时间是采样周期 T 的整数倍，亦即时间延迟常数为 mT。若要输入这样的传递函数模型，只需将传递函数的 `ioDelay` 属性设置成 m，即 `H.ioDelay=m`。

类似于连续系统的零极点模型，离散系统的零极点模型也可以用同样的方法输入，亦即先输入系统的零点和极点，再使用 zpk() 函数就可以输入该模型，注意输入离散系统模型时还应该同时输入采样周期。

例 2-38 已知离散系统的零极点模型为 $H(z)=\dfrac{(z-1/2)(z-1/2+\mathrm{j}/2)(z-1/2-\mathrm{j}/2)}{120(z+1/2)(z+1/3)(z+1/4)(z+1/5)}$，其采样周期为 $T=0.1$ 秒，试将该模型输入 MATLAB 工作空间。

求解 可以用下面的语句输入该系统的数学模型

```
>> z=[1/2; 1/2+1i/2; 1/2-1i/2]; p=[-1/2; -1/3; -1/4; -1/5];
   H=zpk(z,p,1/120,'Ts',0.1)
```

可以得出系统的传递函数模型为 $H(z)=\dfrac{0.0083333(z-0.5)(z^2-z+0.5)}{(z+0.5)(z+0.3333)(z+0.25)(z+0.2)}$。

3. 系统的双线性变换

连续系统和离散系统的传递函数是可以相互转换的，例如令 $s=\dfrac{2(z-1)}{T(z+1)}$，则可以将连续系统传递函数变换成 z 的函数，经过处理就可以直接得到离散系统的传递函数模型，这样的变换又称为双线性变换或 Tustin 变换，这是一种常用的离散化方法。若令 $z=\dfrac{1+Ts/2}{1-Ts/2}$，则可以将离散传递函数模型变换成连续传递函数模型，这样的变换又称为 Tustin 反变换。

除了上述变换方法外，还可以引入其他的变换方法进行连续系统和离散系统的相互变换。例如可以引入基于零阶保持器 ZOH 和一阶保持器 FOH 的变换方法。这样的变换可以通过 MATLAB 提供的 c2d() 函数和 d2c() 直接进行两种模型的相互变换。

```
Gd=c2d(G,T,方法)   % 离散化，方法为 'zoh','foh','tustin' 等
G=d2c(Gd,方法)     % 连续化，无需 T，方法仍如上
```

例 2-39 假设连续系统的数学模型为 $G(s)=\dfrac{1}{(s+2)^3}\mathrm{e}^{-2s}$，选择采样周期为 $T=0.1$ 秒，试得出其等效离散模型。

求解 可以用下面的语句输入该系统的传递函数。

```
>> s=tf('s'); G=1/(s+2)^3; G.ioDelay=2;
```

采用零阶保持器和 Tustin 算法对其离散化，则可以得到下面的结果

```
>> G1=c2d(G,0.1)     % 零阶保持器变换
```

其数学形式为 $G_{\mathrm{ZOH}}(z)=\dfrac{0.0001436z^2+0.0004946z+0.0001064}{z^3-2.456z^2+2.011z-0.5488}z^{-20}$。

```
>> G2=c2d(G,0.1,'tustin') % Tustin 变换
```

其数学形式为

$$G_{\mathrm{Tustin}}(z)=\frac{9.391\times10^{-5}z^3+0.0002817z^2+0.0002817z+9.391\times10^{-5}}{z^3-2.455z^2+2.008z-0.5477}z^{-20}$$

2.5 有理函数的部分分式展开及应用

由于系统的传递函数可以表示成有理函数，所以对有理函数的部分分式展开将在控制系统解析求解中有着重要的作用。本节首先介绍留数概念和求解方法，然后介绍有理函数的部分分式展开方法，并介绍连续、离散传递函数在给定信号激励下的时域响应解析解方法。

2.5.1 留数的概念与计算

在介绍留数概念之前，应该先介绍一下复变函数解析的概念。若函数 $f(z)$ 在复平面的区域内各点处均为单值，且其导数为有限值，则称 $f(z)$ 在复平面内为解析的，这样就将单值函数上不解析的点称为奇点。假设 $z=a$ 为 $f(z)$ 函数上的奇点，若存在一个最小整数 m 使得乘积 $(z-a)^m f(z)$ 在 $z=a$ 点处解析，则称 $z=a$ 为 m 重奇点。

若 $z=a$ 为 $f(z)$ 函数的单奇点，则可以计算出函数在该奇点处的留数为

$$\mathrm{Res}\Big[f(z),z=a\Big]=\lim_{z\to a}(z-a)f(z) \tag{2-5-1}$$

若 $z=a$ 为函数 $f(z)$ 的 m 重奇点，则该点的留数为

$$\mathrm{Res}\Big[f(z),z=a\Big]=\lim_{z\to a}\frac{1}{(m-1)!}\frac{\mathrm{d}^{m-1}}{\mathrm{d}z^{m-1}}\Big[f(z)(z-a)^m\Big] \tag{2-5-2}$$

求取这样的留数方法很简单，假设已知奇点 a 和重数 m，则用下面的 MATLAB 语句自然可以求出相应的留数。

```
c=limit(F*(z-a),z,a)                              % 单奇点
c=limit(diff(F*(z-a)^m,z,m-1)/prod(1:m-1),z,a)    % m 重奇点
```

例 2-40 试求出函数 $f(z)=\dfrac{1}{z^3(z-1)}\sin\left(z+\dfrac{\pi}{3}\right)\mathrm{e}^{-2z}$ 的留数。

求解 对原函数的分析可见，$z=0$ 是三重奇点，$z=1$ 是单奇点，故可以直接使用下面的 MATLAB 语句将这两个奇点处的留数分别求出。

```
>> syms z; f=sin(z+pi/3)*exp(-2*z)/(z^3*(z-1))
   r1=limit(diff(f*z^3,z,2)/prod(1:2),z,0)
   r2=limit(f*(z-1),z,1)
```

可以容易地求出留数为 $r_1=-\dfrac{\sqrt{3}}{4}+\dfrac{1}{2}$, $r_2=\dfrac{1}{2}\mathrm{e}^{-2}\sin 1+\dfrac{\sqrt{3}}{2}\mathrm{e}{-}2\cos 1$。

例 2-41 试求函数 $f(z)=\dfrac{\sin z-z}{z^6}$ 的留数。

求解 乍看该函数很容易认定 $z=0$ 为 6 重奇点，所以用下面的语句很容易就可以求出该点处的留数值为 1/120。

```
>> syms z; f=(sin(z)-z)/z^6;
   limit(diff(f*z^6,z,5)/prod(1:5),z,0)
```

其实这里说 $z=0$ 为 6 重奇点有些太保守，不严格地说，从 $k=1$ 开始尝试，能够使得 $\lim\limits_{z\to 0}\dfrac{\mathrm{d}^{k-1}}{\mathrm{d}z^{k-1}}z^k f(z)<\infty$ 成立的最小的 k 就是 m，可以考虑 $k=2$，可见导数为无穷大，所以再试验更大的 k 值。对此例子来说，m 的最小值为 $m=3$，留数的值仍然为 1/120。试凑更大的 k 值，如 $k=20$，也不会改变求出的留数值。

```
>> syms z; f=(sin(z)-z)/z^6;
   f1=limit(diff(f*z^2,z,1)/prod(1:1),z,0)
   f2=limit(diff(f*z^3,z,2)/prod(1:2),z,0) % 再增加阶次
   f3=limit(diff(f*z^20,z,19)/prod(1:19),z,0) % 再进一步增加阶次
```

则 $f_1=\infty$, $f_2=f_3=1/120$。可见，若选择的 n 值大于或等于奇点的实际重数，则可以正确得到该函数的留数。在一般应用时可选择一个较大的 n 值来求取留数。

例 2-42 试求出 $f(z)=\dfrac{1}{z\sin z}$ 函数的留数。

求解 分析该函数，因为 $\sin z$ 在 $z=0$ 点的收敛速度和 z 是一样的，显然，$z=0$ 点为 $f(z)$ 的二重奇点，这时，相应的留数可以用下面语句求出，其值为 0。

```
>> syms z; f=1/(z*sin(z)); c0=limit(diff(f*z^2,z,1),z,0)
```

进一步分析给定函数 $f(z)$，可以发现该函数在 $z=\pm k\pi$ 处均不解析，其中 k 为正整数，且这些点是原函数的单奇点，由于 MATLAB 的符号运算工具箱并未给出整数的定义，所以这里只能对一些 k 值进行试探，求出它们的留数，最后将结果归纳成所需的公式。

```
>> k=[-4 4 -3 3 -2 2 -1 1]; c=[];
   for kk=k; c=[c,limit(f*(z-kk*pi),z,kk*pi)]; end; c
```

可以得出 $\boldsymbol{c}$ 向量为 $\boldsymbol{c}=\left[-\dfrac{1}{4\pi},\dfrac{1}{4\pi},\dfrac{1}{3\pi},-\dfrac{1}{3\pi},-\dfrac{1}{2\pi},\dfrac{1}{2\pi},\dfrac{1}{\pi},-\dfrac{1}{\pi}\right]$。综上，可以归纳出 $\mathrm{Res}[f(z),z=\pm k\pi]=\pm(-1)^k\dfrac{1}{k\pi}$。

例 2-43 重新考虑例 2-35 中给出的 Z 变换表达式，试用奇点、留数的方式重新表示该结果。

求解 重新计算 Z 变换表达式，得出两个奇点，然后可以根据前面介绍的方法得出两

个奇点处的留数

```
>> syms T s z a K; G=K/s/(s+a); g=ilaplace(G/s);
   Gz=simple((1-z^(-1))*ztrans(g)); [n,d]=numden(Gz);
   x0=solve(d),  % 求出两个奇点
   r1=limit(Gz*(z-x0(1)),z,x0(1)), r2=limit(Gz*(z-x0(2)),z,x0(2))
```

可以得出两个根为 $p_1=1, p_2=\mathrm{e}^{-a}$，并可以求出留数为 $r_1=\dfrac{K}{a}$，$r_2=\dfrac{K(1-\mathrm{e}^{-a})}{a^2}$。由这些结果可以写出原函数的奇点、留数表达式为 $G(z)=\dfrac{K}{a(z-1)}+\dfrac{K(1-\mathrm{e}^{-a})}{a^2(z-\mathrm{e}^{-a})}$。

2.5.2 有理函数的部分分式展开

考虑有理函数

$$G(x)=\frac{B(x)}{A(x)}=\frac{b_1x^m+b_2x^{m-1}+\cdots+b_mx+b_{m+1}}{x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n} \tag{2-5-3}$$

其中，a_i 和 b_i 均为常数。有理函数的互质概念是一非常重要的概念。所谓互质，就是指多项式 $A(x)$ 和 $B(x)$ 没有公约数。对一般给定的两个多项式来说，用手工方式判定多项式互质还是比较困难的，但可以利用 MATLAB 符号运算工具箱中的 gcd() 函数可以直接求出两个多项式的最大公约数。该函数的调用方法为

C=gcd(A,B)

其中，A 和 B 分别表示两个多项式，该函数将得出这两个多项式的最大公约数 C，若得出的 C 为多项式，则两个多项式为非互质的多项式，这时两个多项式可以约简为 A/C 和 B/C。

例 2-44 试判定下面两个多项式是否互质。

$$A(x)=x^4+7x^3+13x^2+19x+20$$
$$B(x)=x^7+16x^6+103x^5+346x^4+655x^3+700x^2+393x+90$$

求解 求解这样的问题可以采用 MATLAB 语言提供的 gcd() 函数完成。

```
>> syms x; A=x^4+7*x^3+13*x^2+19*x+20;
   B=x^7+16*x^6+103*x^5+346*x^4+655*x^3+700*x^2+393*x+90;
   d=gcd(A,B)
```

可见，两个多项式具有最大公约数 $(x+5)$，故两个多项式不是互质的，这两个多项式可以进一步简化为

```
>> A1=simple(A/d), B1=simple(B/d)
```

这样可以得出两个互质的多项式为

$$A_1(x)=x^3+2x^2+3x+4,\ B_1(x)=(x+2)(x+3)^2(x+1)^3$$

若互质多项式 $A(x)=0$ 的根均为相异的值 $-p_i, i=1,2,\cdots,n$，则可以将 $G(x)$ 函数写成下面的部分分式展开形式。

$$G(x)=\frac{r_1}{x+p_1}+\frac{r_2}{x+p_2}+\cdots+\frac{r_n}{x+p_n} \tag{2-5-4}$$

其中 r_i 称为留数，简记作 $\mathrm{Res}[G(-p_i)]$，其值可以由下面的极限式求出。

$$r_i=\mathrm{Res}[G(-p_i)]=\lim_{x\to -p_i} G(s)(x+p_i) \tag{2-5-5}$$

如果分母多项式中含有 $(x+p_i)^k$ 项，亦即 $-p_i$ 为 m 重根，则相对这部分特征根的部分分式展开项可以写成

$$\frac{r_i}{x+p_i}+\frac{r_{i+1}}{(x+p_i)^2}+\cdots+\frac{r_{i+k-1}}{(x+p_i)^k} \tag{2-5-6}$$

这时，r_{i+j-1} 可以用下面的公式直接求出。

$$r_{i+j-1}=\frac{1}{(k-1)!}\lim_{x\to -p_i}\frac{\mathrm{d}^{j-1}}{\mathrm{d}x^{j-1}}\left[G(x)(x+p_i)^k\right],\ j=1,2,\cdots,k \tag{2-5-7}$$

利用 MATLAB 符号运算工具箱中定义的精确求根功能和式 (2-5-5)~(2-5-7) 的留数计算公式，可以立即编写出如下的扩展函数 `residue()`。该函数仍应该置于 `@sym` 目录下，其清单为

```
function f=residue(F,s)
f=sym(0); if nargin==1, syms s; end
[num,den]=numden(F); x0=solve(den); [x,ii]=sort(double(x0));
x0=x0(ii); x=[x0;rand(1)]; kv=find(diff(double(x))~=0); ee=x(kv);
kv=[kv(1); diff(kv(:,1))]; a0=limit(den/s^length(x0),s,inf);
F1=num/(a0*prod(s-x0));% 重组 f(x) 函数的分母，得出新函数 f1(x)
for i=1:length(kv),
   for j=1:kv(i), m=kv(i); s0=ee(i);
      k=limit(diff(F1*(s-s0)^m,s,j-1),s,s0);   % 这里用 f1(x)
      f=f+k/(s-s0)^(m-j+1)/factorial(j-1);
end, end
```

该函数的调用格式为 f=`residue(`F`,`s`)`，其中，F 为有理函数的解析表达式，s 为自变量。返回的结果 f 是部分分式展开的表达式。

例 2-45 考虑 $F(s)=\dfrac{s^3+2s^2+3s+4}{s^6+11s^5+48s^4+106s^3+125s^2+75s+18}$，用解析方式求出其部分分式展开。

求解 用下面的语句可立即得出该函数的部分分式展开式，如下所示，该结果与原例中数值结果完全一致。

```
>> syms s;
   F=(s^3+2*s^2+3*s+4)/(s^6+11*s^5+48*s^4+106*s^3+125*s^2+75*s+18);
   F1=residue(F,s)
```

则可以得出该函数的部分分式展开为

$$F(s)=-\frac{7}{4(s+3)^2}-\frac{17}{8(s+3)}+\frac{2}{s+2}+\frac{1}{2(s+1)^3}-\frac{1}{2(s+1)^2}+\frac{1}{8(s+1)}$$

2.5.3 基于部分分式展开的 Laplace 变换

由前面得出的部分分式展开式，利用 Laplace 反变换的公式

$$\mathscr{L}\left[\frac{r}{(s+a)^m}\right]=\frac{rt^{m-1}}{(m-1)!}\mathrm{e}^{-at}$$

则可以由部分分式展开的式中容易地写出其反变换。

例 2-46 利用例 2-45 中给出的部分分式展开的结论，则可以直接写出该函数的 Laplace 反变换为 $f(t)=\left(-\frac{7}{4}t-\frac{17}{8}\right)\mathrm{e}^{-3t}-2\mathrm{e}^{-2t}+\left(\frac{1}{4}t^2-\frac{1}{2}t+\frac{1}{8}\right)\mathrm{e}^{-t}$。

对离散系统来说，分母多项式有重根，则可以根据下面的式子进行部分分式展开[4]

$$\mathscr{Z}^{-1}\left[\frac{q}{(z^{-1}-p)^m}\right]=\frac{(-1)^m q}{(m-1)!\,(-p)^{n+m}}(n+1)(n+2)\cdots(n+m-1) \tag{2-5-8}$$

当然，若使用 `laplace()` 函数和 `ilaplace()` 函数，则无需实现对其进行部分分式展开，利用该函数自身的功能就能直接得出所需的展开式。

2.5.4 有理式部分分式展开在控制系统中的应用

掌握了有理函数部分分式展开的求解方法，则可以通过该方法求解连续、离散传递函数模型的时域响应解析解方法。

1. 连续系统响应解析解

假设连续系统的传递函数可以写成 $G(s)$，若系统输入信号的 Laplace 变换为 $U(s)$，则输出信号的 Laplace 变换式可以写成 $Y(s)=G(s)U(s)$，这样系统响应的解析解可以由 Laplace 反变换直接求解，即 $y(t)=\mathscr{L}^{-1}[Y(s)]$。

例 2-47 假设系统的传递函数模型为 $G(s)=\dfrac{s^3+7s^2+3s+4}{s^4+7s^3+17s^2+17s+6}$，若系统的输入信号为 $u(t)=2+2\mathrm{e}^{-3t}\sin(2t)$，试求出系统响应的解析解。

求解 用下面的语句首先计算出输入信号的 Laplace 变换，然后计算出输出信号，最后可以通过 Laplace 反变换的方法得出系统响应的解析解。

```
>> syms s t;
   G=(s^3+7*s^2+3*s+4)/(s^4+7*s^3+17*s^2+17*s+6);
   u=2+2*exp(-3*t)*sin(2*t); U=laplace(u); y=ilaplace(G*U)
```

得出的解析解为

$$y(t)=\frac{4}{3}-\frac{31}{12}\mathrm{e}^{-3t}-\frac{23}{20}\mathrm{e}^{-3t}\cos 2t+\left(6-\frac{21}{4}t\right)\mathrm{e}^{-t}-\frac{18}{5}\mathrm{e}^{-2t}-\frac{103}{40}\mathrm{e}^{-3t}\sin 2t$$

若已知传递函数含有时间延迟，即 $\widehat{G}(s)=G(s)\mathrm{e}^{-2s}$，则利用上述方法仍然能求出系统响应的解析解

```
>> G1=G*exp(-2*s); Y1=G1*U; y1=ilaplace(Y1)
```

这样可以写出系统响应的解析解为

$$\begin{aligned}y(t)=&\left[\frac{4}{3}-\frac{31}{12}\mathrm{e}^{-3t+6}-\frac{18}{5}\mathrm{e}^{-2t+4}+\frac{33}{2}\mathrm{e}^{-t+2}-\frac{21}{4}\mathrm{e}^{-t+2}t\right.\\&\left.-\frac{23}{20}\mathrm{e}^{-3t+6}\cos(2t-4)-\frac{103}{40}\mathrm{e}^{-3t+6}\sin(2t-4)\right]\mathrm{Heaviside}(t-2)\end{aligned}$$

2. 离散系统响应解析解

如果离散系统的传递函数模型为 $H(z)$，且可以求出输入信号的 Z 变换为 $U(z)$，则输出信号的 Z 变换为 $Y(z)=H(z)U(z)$，这样输出信号的时域响应解析解可以通过 Z 反变换直接求出，即 $y(k)=\mathscr{Z}^{-1}[Y(z)]$。

例 2-48 假设系统的传递函数模型为 $G(z)=\dfrac{(z-1/3)}{(z-1/2)(z-1/4)(z+1/5)}$，若输入信号为单位阶跃信号，试求出此离散系统的阶跃响应解析解。

求解 可以通过如下的语句直接求取系统阶跃响应的解析解。

```
>> syms z t; H=(z-1/3)/(z-1/2)/(z-1/4)/(z+1/5);
   U=ztrans(sym(1)); Y=H*U; y=iztrans(Y)
```

故系统的解析解为 $y(n)=\dfrac{800}{567}\left(-\dfrac{1}{5}\right)^n-\dfrac{40}{21}\left(\dfrac{1}{2}\right)^n+\dfrac{40}{27}-\dfrac{80}{81}\left(\dfrac{1}{4}\right)^n$。若系统的采样周期为 T，则原系统的解析解还可以写成

$$y(kT)=\frac{800}{567}\left(-\frac{1}{5}\right)^{kT}-\frac{40}{21}\left(\frac{1}{2}\right)^{kT}+\frac{40}{27}-\frac{80}{81}\left(\frac{1}{4}\right)^{kT}$$

例 2-49 假设某离散系统的传递函数及输入信号如下给出，试求输出信号的解析解。

$$H(z)=\frac{z(5z-2)}{(z-1/2)^3(z-1/3)},\quad 且\ u(k)=\begin{cases}1, & k=0,2,4,\cdots\\-1, & k=1,3,5,\cdots\end{cases}$$

求解 显然，输入信号的解析表达式可以统一写成 $u(k)=\cos k\pi$，这样系统输出信号的解析解可以由下面的语句直接求出

```
>> syms z k; H=z*(5*z-2)/(z-1/2)^3/(z-1/3);
   u=cos(k*pi); U=ztrans(u); y=iztrans(U*H)
```

这时输出信号的解析解为 $y(k)=\left(-\dfrac{176}{9}+\dfrac{26k}{3}+2k^2\right)\left(\dfrac{1}{2}\right)^k+\dfrac{14}{9}(-1)^k+18\left(\dfrac{1}{3}\right)^k$。

2.6 控制系统结构图化简

前面介绍了传递函数、状态方程及零极点模型的输入，但控制系统的模型输入并不总是这样简单，一般的控制系统均需要由若干个子模型进行互连，才能构造出来。所以在本节中将介绍子模块的互连及总系统模型的获取。首先介绍三类典型的连接结构：串联、并联和反馈连接，然后介绍模块输入、输出从一个节点移动到另一个节点所必需的等效变换，最后介绍复杂系统的等效变换和化简。本节介绍的内容包括数值解和解析模型推导。

2.6.1 控制系统的典型连接结构

两个模块 $\boldsymbol{G}_1(s)$ 和 $\boldsymbol{G}_2(s)$ 的串联连接如图 2-6 (a) 所示，在这样的结构下，输入信号 $\boldsymbol{u}(t)$ 流过第一个模块 $\boldsymbol{G}_1(s)$，而模块 $\boldsymbol{G}_1(s)$ 的输出信号输入到第二个模块 $\boldsymbol{G}_2(s)$，该模块的输出 $\boldsymbol{y}(t)$ 是整个系统的输出。在串联连接下，整个系统的传递函数为 $\boldsymbol{G}(s)=\boldsymbol{G}_2(s)\boldsymbol{G}_1(s)$。对单变量系统来说，这两个模块 $\boldsymbol{G}_1(s)$ 和 $\boldsymbol{G}_2(s)$ 是可以互换的，亦即 $\boldsymbol{G}_1\boldsymbol{G}_2=\boldsymbol{G}_2\boldsymbol{G}_1$，对多变量系统来说，一般不具备这样的关系。

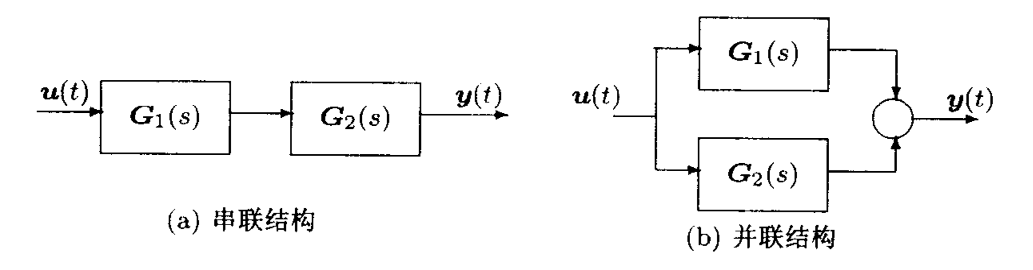


图 2-6 系统的串联、并联结构

由上面的理论可以看出，若两个模块同样用传递函数描述，则可以由有理函数相乘的方法得出总模型，但如果有一个用传递函数描述，另一个用状态方程表示，则在总模型求取上有一定的麻烦，需要在求解总模型前转换成一致的模型。MATLAB 的控制系统工具箱成功地解决了这样的问题，若已知两个子系统模型 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$，则串联结构总的系统模型可以统一由 `G = G2*G1` 求出。

两个模块 $\boldsymbol{G}_1(s)$ 和 $\boldsymbol{G}_2(s)$ 的典型并联连接结构如图 2-6 (b) 所示，其中这两个模块在共同的输入信号 $\boldsymbol{u}(t)$ 激励下，产生两个输出信号，而系统总的输出信号 $\boldsymbol{y}(t)$ 是这两个输出信号的和。并联系统的传递函数总模型为 $\boldsymbol{G}(s)=\boldsymbol{G}_1(s)+\boldsymbol{G}_2(s)$。

在 MATLAB 下，若已知两个子系统模型 G_1 和 G_2，则并联结构总的系统模型可以统一由 $G = G_1 + G_2$ 求出。

两个模块 $G_1(s)$ 和 $G_2(s)$ 的两种反馈连接结构分别如图 2-7 (a)、(b) 所示。前一种反馈结构称为正反馈结构，后一种称为负反馈结构。反馈系统总的模型为

$$\begin{aligned}&\text{正反馈:}\quad \boldsymbol{G}(s) = \boldsymbol{G}_1(s)[\boldsymbol{I} - \boldsymbol{G}_1(s)\boldsymbol{G}_2(s)]^{-1}\\&\text{负反馈:}\quad \boldsymbol{G}(s) = \boldsymbol{G}_1(s)[\boldsymbol{I} + \boldsymbol{G}_1(s)\boldsymbol{G}_2(s)]^{-1}\end{aligned} \tag{2-6-1}$$

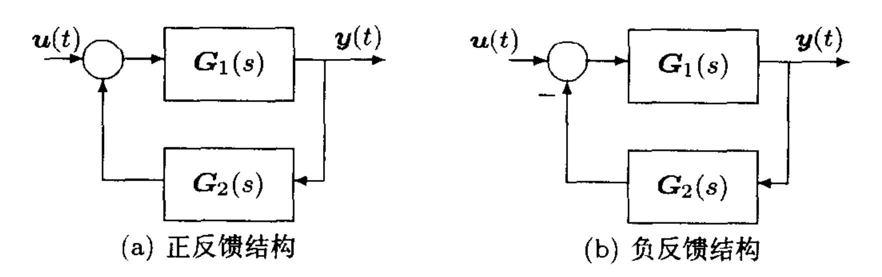


图 2-7　系统的反馈连接结构

在 MATLAB 环境中直接能使用 `G=G1*inv(eye(size(G1))+G1*G2)` 这样的语句求取总系统模型，但这样得出的模型阶次可能高于实际的阶次，需要用 `minreal()` 函数求取得出模型的最小实现形式。此外还可以使用 MATLAB 控制系统工具箱中提供的 `feedback()` 函数求取总模型，该函数的调用格式如下：

```
G=feedback(G1,G2);  % 负反馈连接
G=feedback(G1,G2,1);  % 正反馈连接
```

MATLAB 提供的 `feedback()` 函数只能用于 G_1 和 G_2 为具体参数给定的模型，通过适当的扩展，就可以编写一个能够处理符号运算的 `feedback()` 函数。

```
function H=feedback(G1,G2,key)
if nargin==2; key=-1; end, H=G1/(sym(1)-key*G1*G2); H=simple(H);
```

若将其放置在 MATLAB 路径下某个目录的 `@sym` 子目录下，例如在 work 目录下建立一个 `@sym` 子目录，将该文件置于子目录下，则可以直接处理符号模型的化简问题，而不影响原来数值型 `feedback()` 函数的正常调用。

例 2-50　考虑如图 2-8 所示的典型反馈控制系统框图，假设各个子传递函数模型为

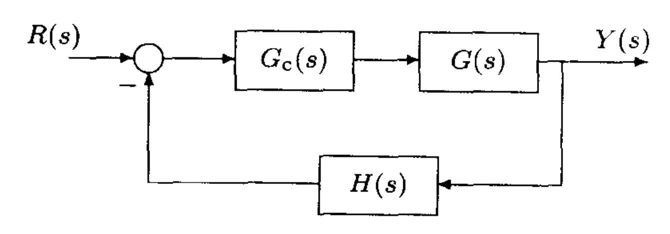


图 2-8　典型反馈控制系统方框图

$$G(s) = \frac{12s^3 + 24s^2 + 12s + 20}{2s^4 + 4s^3 + 6s^2 + 2s + 2}$$

$$G_c(s) = \frac{5s+3}{s},\quad H(s) = \frac{1000}{s+1000}$$

试求出其闭环传递函数模型。

求解 输入了各个子传递函数，则可以通过下面的语句将总模型用 MATLAB 求出。

```
>> s=tf('s'); G=(12*s^3+24*s^2+12*s+20)/(2*s^4+4*s^3+6*s^2+2*s+2);
   Gc=(5*s+3)/s; H=1000/(s+1000);
   GG=feedback(G*Gc,H)    % 求取并显示负反馈系统的传递函数模型
```

其数学表示为

$$G_{闭}(s)=\frac{60s^5+60156s^4+156132s^3+132136s^2+136060s+60000}{2s^6+2004s^5+64006s^4+162002s^3+134002s^2+138000s+60000}$$

2.6.2 节点移动时的等效变换

在复杂结构图化简中，经常需要将某个支路的输入点从一个节点移动到另一个节点上，例如在图 2-9 中给出的方框图中，比较难处理的地方是 $G_2(s)$，$G_3(s)$ 和 $H_2(s)$ 构成的回路，应该将 $H_2(s)$ 模块的输入端从 A 点等效移动到系统的输出端 $Y(s)$，这就需要对这样的移动导出等效的变换。

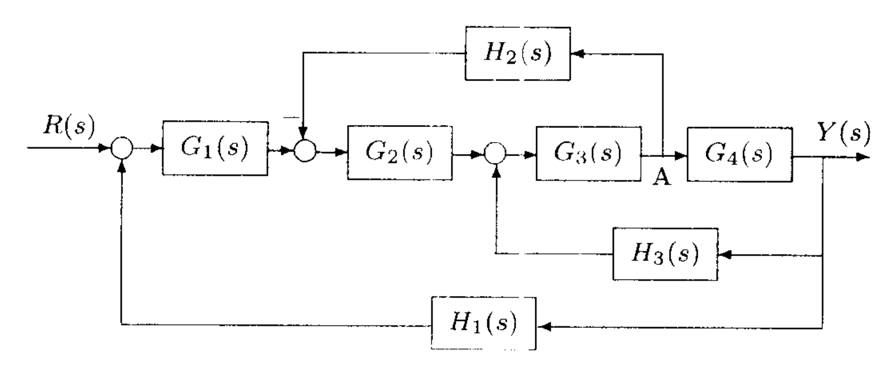


图 2-9 控制系统的方框图

图 2-10 (a)、(b) 中分别定义了两种常用的节点移动方式：节点前向移动和后向移动。在图 2-10 (a) 中，若想将 $G_2(s)$ 支路的起始点从 A 点移动到 B 点，则需要将新的 $G_2(s)$ 支路乘以 $G_1(s)$ 模型，这样的移动称为节点的前向移动；而图 2-10 (b) 中，若想将 $G_2(s)$ 支路的起始点从 B 点移动到 A 点，则需要将新的 $G_2(s)$ 支路除以 $G_1(s)$ 模型，这样的移动称为节点的后向移动。如果用 MATLAB 表示，则前向移动后新的支路模型变成了 `G2*G1`，而后向移动后该支路变成了 `G2/G1`，或 `G2*inv(G1)`。

2.6.3 复杂系统模型的简化

利用前面给出的等效变换方法不难对更复杂的系统进行化简，本节中将通过例子来演示这样的化简。

例 2-51 假设系统的方框图模型如图 2-9 所示，为方便对其处理，应该将 $H_2(s)$ 模块的输入端从 A 点等效移动到系统的输出端 $Y(s)$，如图 2-11 所示。

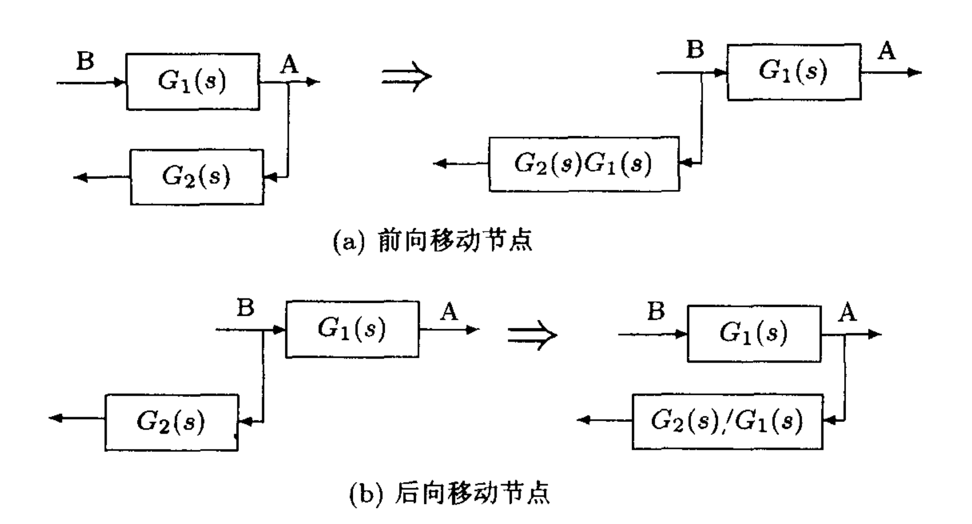


图 2-10　节点移动等效变换

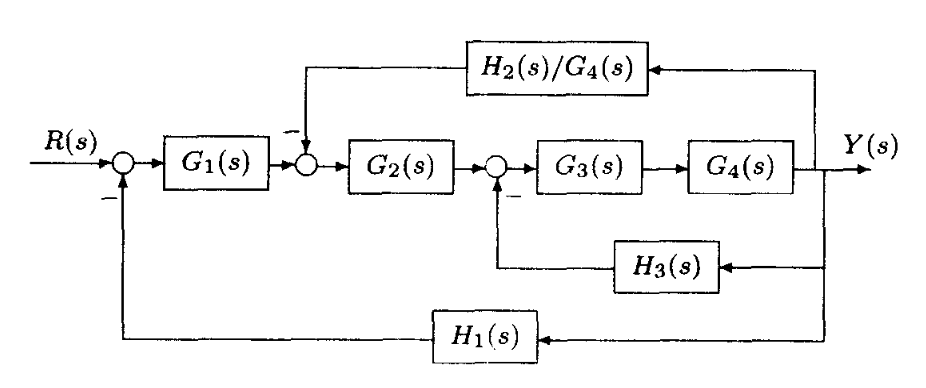


图 2-11　变换后的方框图

得到了这样的化简框图后，可以清晰地看出：最内层的闭环是由 $G_3(s)$, $G_4(s)$ 的串联为前向通路，以 $H_3(s)$ 为反馈通路构成的负反馈结构，利用前面介绍的知识可以马上得出这个子模型，该子模型与 $G_2(s)$ 串联又构成了第二层回路的前向通路，它与变换后的 $H_2(s)/G_4(s)$ 通路构成负反馈结构，结果再与 $G_1(s)$ 串联，与 $H_1(s)$ 构成负反馈结构。通过这样的逐层变换就可容易地求出总的系统模型。上面的分析可以用下面的 MATLAB 语句实现，从而得出总的系统模型。

```
>> syms G1 G2 G3 G4 H1 H2 H3 % 定义各个子模块为符号变量
   c1=feedback(G4*G3,H3);     % 最内层闭环模型
   c2=feedback(c1*G2,H2/G4); % 第二层闭环模型
   G=feedback(c2*G1,H1); pretty(G) % 总系统模型
```

得出结果的数学表示形式为 $G(s)=\dfrac{G_2G_4G_3G_1}{1+G_4G_3H_3+G_3G_2H_2+G_2G_4G_3G_1H_1}$。

例 2-52　考虑如图 2-12 所示的电机拖动系统模型，该系统有双输入，给定输入 $r(t)$ 和负载输入 $M(t)$，利用 MATLAB 符号运算工具箱可以推导出系统的传递函数矩阵。

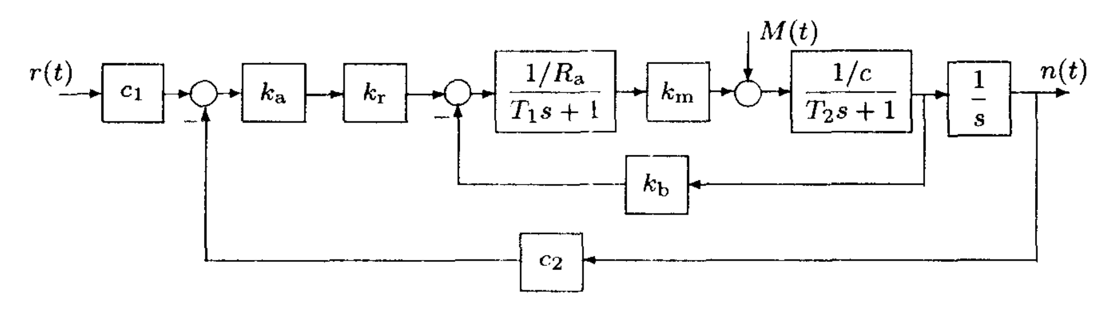


图 2-12 双输入系统方框图

先考虑输入 $r(t)$ 输入信号单独激励系统，则能用最简单的方式得出传递函数模型

```
>> syms Ka Kr c1 c2 c Ra T1 T2 Km Kb s   % 申明符号变量
   Ga=feedback(1/Ra/(T1*s+1)*Km*1/c/(T2*s+1),Kb);
   G1=c1*feedback(Ka*Kr*Ga/s,c2); G1=collect(G1,s)
```

这样可以得出子传递函数，显示暂略，后面将用数学形式给出。

若 $M(t)$ 输入信号单独作用时，对原系统结构稍微改动一下，则可以得出如图 2-13 所示的新框图，故用下面的语句能直接计算出传递函数模型

```
>> G2=feedback(1/c/(T2*s+1)/s, Km/Ra/(T1*s+1)*(Kb*s+c2*Ka*Kr));
   G2=collect(simplify(G2),s)
```

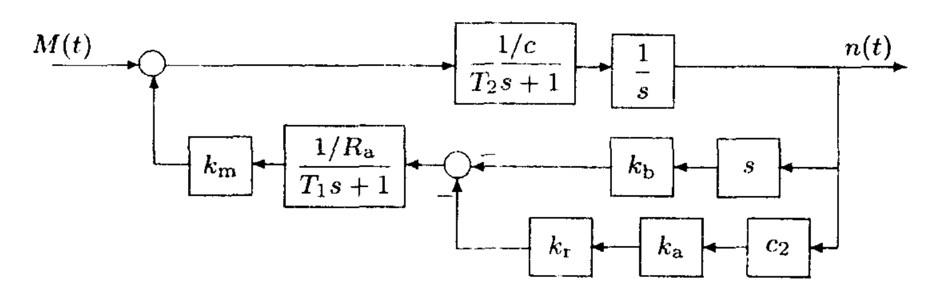


图 2-13 $M(t)$ 单独激励时等效系统方框图

综上所述，可以用 MATLAB 语言推导出系统的传递函数矩阵为

$$\boldsymbol{G}^{\mathrm{T}}(s)=\begin{bmatrix}\dfrac{c_1k_\mathrm{m}k_\mathrm{a}k_\mathrm{r}}{R_\mathrm{a}cT_1T_2s^3+(R_\mathrm{a}cT_1+R_\mathrm{a}cT_2)s^2+(k_\mathrm{m}k_\mathrm{b}+R_\mathrm{a}c)s+k_\mathrm{a}k_\mathrm{r}k_\mathrm{m}c_2}\\[2ex]\dfrac{(T_1s+1)R_\mathrm{a}}{cR_\mathrm{a}T_2T_1s^3+(cR_\mathrm{a}T_1+cR_\mathrm{a}T_2)s^2+(k_\mathrm{b}k_\mathrm{m}+cR_\mathrm{a})s+k_\mathrm{m}nc_2k_\mathrm{a}k_\mathrm{r}}\end{bmatrix}$$

2.7 习题与思考题

1 试求出如下极限。

① $\lim\limits_{x\to\infty}(3^x+9^x)^{1/x}$， ② $\lim\limits_{x\to\infty}\dfrac{(x+2)^{x+2}(x+3)^{x+3}}{(x+5)^{2x+5}}$

③ $\lim\limits_{x\to 0}\dfrac{\ln(1+x)\ln(1-x)-\ln(1-x^2)}{x^4}$

2 试求下面的双重极限。

① $\lim\limits_{\substack{x\to -1\\ y\to 2}}\dfrac{x^2y+xy^3}{(x+y)^3}$， ② $\lim\limits_{\substack{x\to 0\\ y\to 0}}\dfrac{xy}{\sqrt{xy+1}-1}$， ③ $\lim\limits_{\substack{x\to 0\\ y\to 0}}\dfrac{1-\cos\left(x^2+y^2\right)}{\left(x^2+y^2\right)\mathrm{e}^{x^2+y^2}}$

3 求出下面函数的导数。

① $y(x)=\sqrt{x\sin x\sqrt{1-\mathrm{e}^x}}$， ② $y(t)=\sqrt{\dfrac{(x-1)(x-2)}{(x-3)(x-4)}}$

③ $\arctan\dfrac{y}{x}=\ln(x^2+y^2)$， ④ $y(x)=-\dfrac{1}{na}\ln\dfrac{x^n+a}{x^n}$, $n>0$

4 试求出 $y=\dfrac{1-\sqrt{\cos ax}}{x\left(1-\cos\sqrt{ax}\right)}$ 函数的 10 阶导数。

5 在高等数学中，求解分子和分母均同时为 0 或 ∞ 的分式极限时可使用 L'Hôpital 法则，即对分子分母分别求导数，再由比值得出。试用该法则求习题 1 的②、③中的极限问题，并和直接求出的极限结果相比较。

6 已知参数方程 $\begin{cases} x=\ln\cos t\\ y=\cos t-t\sin t\end{cases}$，试求出 $\dfrac{\mathrm{d}y}{\mathrm{d}x}$ 和 $\left.\dfrac{\mathrm{d}^2y}{\mathrm{d}x^2}\right|_{t=\pi/3}$。

7 假设 $u=\cos^{-1}\sqrt{\dfrac{x}{y}}$，试验证 $\dfrac{\partial^2 u}{\partial x\,\partial y}=\dfrac{\partial^2 u}{\partial y\,\partial x}$。

8 设 $\begin{cases} xu+yv=0\\ yu+xv=1\end{cases}$，试求解 $\dfrac{\partial^2 u}{\partial x\,\partial y}$。

9 试求解下面的不定积分问题。

① $I(x)=-\displaystyle\int\dfrac{3x^2+a}{x^2\left(x^2+a\right)^2}\mathrm{d}x$， ② $I(x)=\displaystyle\int\dfrac{\sqrt{x(x+1)}}{\sqrt{x}+\sqrt{1+x}}\mathrm{d}x$

③ $I(x)=\displaystyle\int x\mathrm{e}^{ax}\cos bx\mathrm{d}x$， ④ $I(t)=\displaystyle\int \mathrm{e}^{ax}\sin bx\sin cx\mathrm{d}x$

10 试求出下面的定积分或无穷积分。

① $I=\displaystyle\int_0^\infty\dfrac{\cos x}{\sqrt{x}}\mathrm{d}x$， ② $I=\displaystyle\int_0^1\dfrac{1+x^2}{1+x^4}\mathrm{d}x$

11 试对下面函数进行 Fourier 幂级数展开。

① $f(x)=(\pi-|x|)\sin x$, $-\pi\leqslant x<\pi$； ② $f(x)=\mathrm{e}^{|x|}$, $-\pi\leqslant x<\pi$；

③ $f(x)=\begin{cases} 2x/l, & 0<x<l/2\\ 2(l-x)/l, & l/2<x<l\end{cases}$，且 $l=\pi$

12 试求出下面函数的 Taylor 幂级数展开。

① $\displaystyle\int_0^x \frac{\sin t}{t}\mathrm{d}t$， ② $\ln\left(\dfrac{1+x}{1-x}\right)$， ③ $\ln\left(x+\sqrt{1+x^2}\right)$， ④ $(1+4.2x^2)^{0.2}$

⑤ $\mathrm{e}^{-5x}\sin(3x+\pi/3)$ 分别关于 $x=0$、$x=a$ 的幂级数展开

⑥ 对 $f(x,y)=\dfrac{1-\cos\left(x^2+y^2\right)}{\left(x^2+y^2\right)\mathrm{e}^{x^2+y^2}}$ 关于 $x=1,y=0$ 进行二维 Taylor 幂级数展开

13 对下列的函数 $f(t)$ 进行 Laplace 变换。

① $f(t)=\dfrac{\sin\alpha t}{t}$， ② $f(t)=t^5\sin\alpha t$， ③ $f(t)=t^8\cos\alpha t$， ④ $f(t)=t^6\mathrm{e}^{\alpha t}$

⑤ $f(t)=5\mathrm{e}^{-at}+t^4\mathrm{e}^{-at}+8\mathrm{e}^{-2t}$， ⑥ $f(t)=\mathrm{e}^{\beta t}\sin(\alpha t+\theta)$， ⑦ $f(t)=\mathrm{e}^{-12t}+6\mathrm{e}^{9t}$

14 对下面的 $F(s)$ 式进行 Laplace 反变换。

① $F(s)=\dfrac{1}{\sqrt{s^2(s^2-a^2)(s+b)}}$， ② $F(s)=\sqrt{s-a}-\sqrt{s-b}$， ③ $F(s)=\ln\dfrac{s-a}{s-b}$

④ $F(s)=\dfrac{1}{\sqrt{s}(s+a)}$， ⑤ $F(s)=\dfrac{3a^2}{s^3+a^3}$， ⑥ $F(s)=\dfrac{(s-1)^8}{s^7}$

⑦ $F(s)=\ln\dfrac{s^2+a^2}{s^2+b^2}$， ⑧ $F(s)=\dfrac{s^2+3s+8}{\prod_{i=1}^{8}(s+i)}$， ⑨ $F(s)=\dfrac{1}{2}\dfrac{s+\alpha}{s-\alpha}$

15 Laplace 变换的一个重要应用是求解常系数线性微分方程，可以利用当函数具有零初始值和各阶导数的零初始值下 $\mathscr{L}[\mathrm{d}^n f(t)/\mathrm{d}t^n]=s^n\mathscr{L}[f(t)]$ 这一性质对微分方程进行 Laplace 变换的方法求解微分方程。试使用这样的方法求解下面的微分方程 $y''(t)+3y'(t)+2y(t)=\mathrm{e}^{-t}$，$y(0)=y'(0)=0$。

16 假设描述系统的常微分方程为 $y^{(3)}(t)+10\ddot{y}(t)+32\dot{y}(t)+32y(t)=6u^{(3)}(t)+4\ddot{u}(t)+2u(t)+2\dot{u}(t)$，请用 MATLAB 语言表示该方程的数学模型。该模型的零极点模型如何求取？由微分方程模型能否直接写出系统的传递函数模型？

17 请将下面的系统模型输入到 MATLAB 环境

① $G(s)=\dfrac{8(s+1-\mathrm{j})(s+1+\mathrm{j})}{s^2(s+5)(s+6)(s^2+1)}$， ② $G(s)=\dfrac{s^2+5s+6}{[(s+1)^2+1](s+2)(s+4)}$

18 试求出下面函数的 Fourier 变换，对得出的结果再进行 Fourier 反变换，观察是否能得出原函数。

① $f(x)=x^2(3\pi-2|x|)$, $0\leqslant x\leqslant 2\pi$， ② $f(t)=t^2(t-2\pi)^2$, $0\leqslant t\leqslant 2\pi$

③ $f(t)=\mathrm{e}^{-t^2}$，$-l\leqslant t\leqslant l$， ④ $f(t)=t\mathrm{e}^{-|t|}$, $-\pi\leqslant t\leqslant\pi$

19 试证明 $\cos\theta+\cos 2\theta+\cdots+\cos n\theta=\dfrac{\sin(n\theta/2)\cos[(n+1)\theta/2]}{\sin\theta/2}$。

20 试求出下面函数的 Fourier 正弦和余弦变换，并用 Fourier 正弦、余弦反变换对得出的结果进行处理，观察是否能还原成原始函数。

① $f(t)=\mathrm{e}^{-t}\ln t$， ② $f(x)=\dfrac{\cos x^2}{x}$， ③ $f(x)=\ln\dfrac{1}{\sqrt{1+x^2}}$

21 试求下面函数的离散 Fourier 正弦、余弦变换。

① $f(x)=\mathrm{e}^{kx}$, ② $f(x)=x^3$

22 请将下述时域序列函数 $f(kT)$ 进行 Z 变换，并对结果进行反变换检验。

① $f(kT)=\cos(kaT)$, ② $f(kT)=(kT)^2\mathrm{e}^{-akT}$, ③ $f(kT)=\dfrac{1}{a}(akT-1+\mathrm{e}^{-akT})$

④ $f(kT)=\mathrm{e}^{-akT}-\mathrm{e}^{-bkT}$, ⑤ $f(kT)=\sin(\alpha kT)$,⑥ $f(kT)=1-\mathrm{e}^{-akT}(1+akT)$

23 已知下述各个 Z 变换表达式 $F(z)$，试对它们分别进行 Z 反变换。

① $F(z)=\dfrac{10z}{(z-1)(z-2)}$, ② $F(z)=\dfrac{z^2}{(z-0.8)(z-0.1)}$, ③ $F(z)=\dfrac{z}{(z-a)(z-1)^2}$

④ $F(z)=\dfrac{z^{-1}(1-\mathrm{e}^{-aT})}{(1-z^{-1})(1-z^{-1}\mathrm{e}^{-aT})}$, ⑤ $F(z)=\dfrac{Az[z\cos\beta-\cos(\alpha T-\beta)]}{z^2-2z\cos(\alpha T)+1}$

24 已知某信号的 Laplace 变换为 $\dfrac{b}{s^2(s+a)}$，试求其 Z 变换，并验证结果。

25 请将下面的传递函数模型输入到 MATLAB 环境。

① $H(z)=\dfrac{5(z-0.2)^2}{z(z-0.4)(z-1)(z-0.9)+0.6}$, $T=0.1$ 秒

② $H(z^{-1})=\dfrac{(z^{-1}+3.2)(z^{-1}+2.6)}{z^{-5}(z^{-1}-8.2)}$, $T=0.05$ 秒

26 已知某系统的差分方程模型为 $y(k+2)+y(k+1)+0.16y(k)=u(k-1)+2u(k-2)$，试将其输入到 MATLAB 工作空间。

27 试求出 $f(x)=\dfrac{x^2+4x+3}{x^5+4x^4+3x^3+2x^2+5x+2}\mathrm{e}^{-5x}$ 的奇点、奇点重数及各个奇点处的留数。

28 从下面给出的典型反馈控制系统结构子模型中，求出总系统的状态方程与传递函数模型，并得出各个模型的零极点模型表示。

① $G(s)=\dfrac{211.87s+317.64}{(s+20)(s+94.34)(s+0.17)}$, $G_{\mathrm{c}}(s)=\dfrac{169.6s+400}{s(s+4)}$, $H(s)=\dfrac{1}{0.01s+1}$

② $G(z^{-1})=\dfrac{35786.7z^{-1}+108444}{(z^{-1}+4)(z^{-1}+20)(z^{-1}+74)}$, $G_{\mathrm{c}}(z^{-1})=\dfrac{1}{z^{-1}-1}$, $H(z^{-1})=\dfrac{1}{0.5z^{-1}-1}$

③ $G(s)=\dfrac{K_{\mathrm{m}}J}{Js^2+Bs+K_{\mathrm{r}}}$, $G_{\mathrm{c}}(s)=\dfrac{L_{\mathrm{q}}}{L_{\mathrm{q}}s+R_{\mathrm{q}}}$, $H(s)=sK_{\mathrm{v}}$

29 假设系统的对象模型，并定义一个 PID 控制器

$$G(s)=\frac{10}{(s+1)^3},\quad G_{\mathrm{PID}}(s)=0.48\left(1+\frac{1}{1.814s}+\frac{0.4353s}{1+0.04353s}\right)$$

这个控制器与对象模型进行串联连接，假定整个闭环系统是由单位负反馈构成的，请求出闭环系统的传递函数模型，并求出该模型的各种状态方程的标准型实现和零极点模型。

30 假设多变量系统和控制器如下给出

$$G(s)=\begin{bmatrix}\dfrac{-0.252}{(1+3.3s)^3(1+1800s)} & \dfrac{0.43}{(1+12s)(1+1800s)}\\ \dfrac{-0.0435}{(1+25.3s)^3(1+360s)} & \dfrac{0.097}{(1+12s)(1+360s)}\end{bmatrix},\ G_c(s)=\begin{bmatrix}-10 & 77.5\\ 0 & 50\end{bmatrix}$$

试求出单位负反馈下闭环系统的传递函数矩阵模型，并得出状态方程模型。

31 已知系统的方框图如图 2-14 所示，试推导出从输入信号 $r(t)$ 到输出信号 $y(t)$ 的总系统模型。

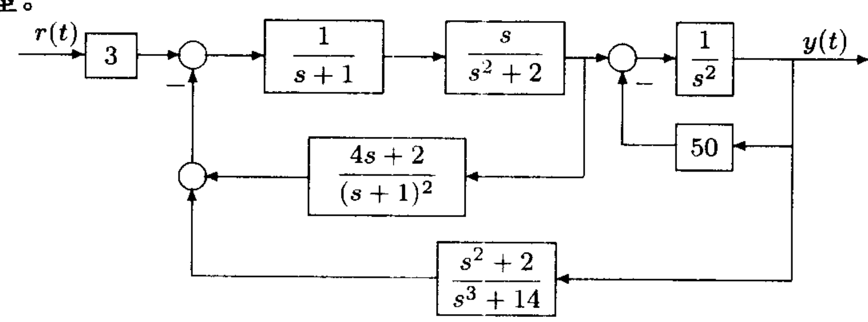


图 2-14 某线性系统结构图

32 已知系统的方框图如图 2-15 所示，试推导出从输入信号 $r(t)$ 到输出信号 $y(t)$ 的总系统模型。

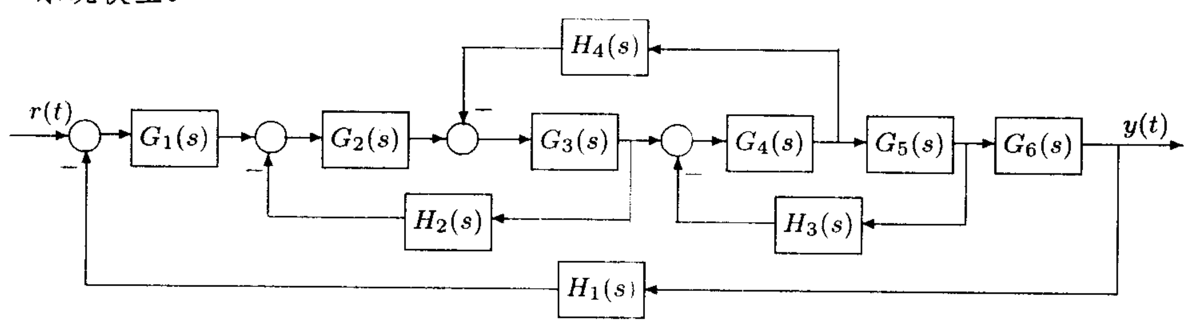


图 2-15 某线性系统结构图

33 在 Simulink 等软件环境出现之前，为衡量仿真工具的优劣曾出现了各种各样的基准测试模型，F-14 战斗机模型就是其中之一[5]，该系统框图如图 2-16 所示，该系统共有两路输入信号，其向量表示为 $\boldsymbol{u}=[n(t),\alpha_c(t)]^{\mathrm{T}}$，其中 $n(t)$ 为单位方差的白噪声信号，而 $\alpha_c(t)=K\beta(\mathrm{e}^{-\gamma t}-\mathrm{e}^{-\beta t})/(\beta-\gamma)$ 为攻击角度命令输入信号，这里 $K=\alpha_{c_{\max}}\mathrm{e}^{\gamma t_m}$，且 $\alpha_{c_{\max}}=0.0349$, $t_m=0.025$, $\beta=426.4352$, $\gamma=0.01$，整个系统的输出有三路信号，$\boldsymbol{y}(t)=[N_{Z_p}(t),\alpha(t),q(t)]^{\mathrm{T}}$，这里 $N_{Z_p}(t)$ 信号定义为 $N_{Z_p}(t)=\dfrac{1}{32.2}[-\dot{w}(t)+U_0q(t)+22.8\dot{q}(t)]$，已知系统中各个模块的参数为：

$\tau_a=0.05$, $\sigma_{wG}=3.0$, $a=2.5348$, $b=64.13$

$V_{\tau_0}=690.4, \sigma_\alpha=5.236\times10^{-3}, Z_b=-63.9979, M_b=-6.8847$

$$U_0=689.4, Z_w=-0.6385, M_q=-0.6571, M_w=-5.92\times10^{-3}$$

$$\omega_1 = 2.971,\ \omega_2 = 4.144,\ \tau_s = 0.10,\ \tau_\alpha = 0.3959$$

$$K_Q = 0.8156,\ K_\alpha = 0.6770,\ K_f = -3.864,\ K_F = -1.745$$

试采用系统互联的方法得出系统的闭环系统模型。

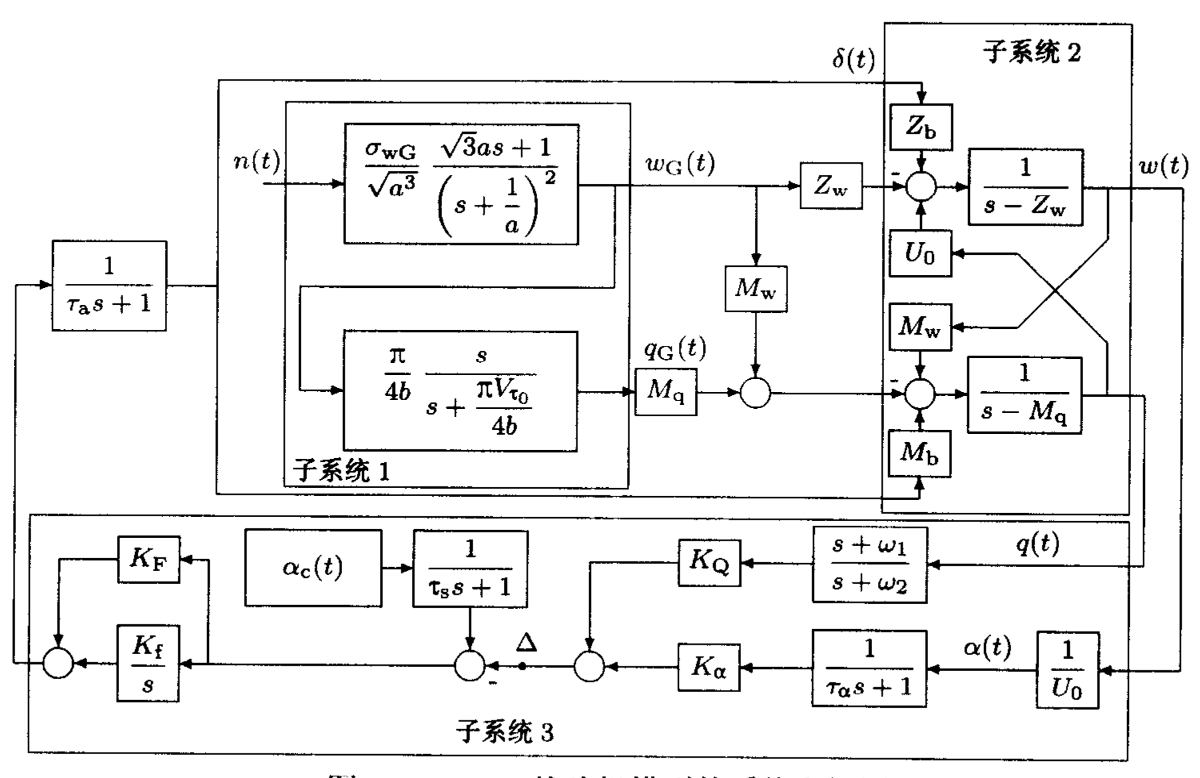


图 2-16 F-14 战斗机模型的系统方框图

参考文献

[1] Munro N. Multivariable control 1: the inverse Nyquist array design method, In: Lecture notes of SERC vacation school on control system design [M]. UMIST, Manchester, 1989

[2] 《数学手册》编写组. 数学手册 [M]. 北京: 人民教育出版社, 1979

[3] Kuo B C. Digital control systems [M]. New York: Holt, Rinehart and Winston. Inc, 1980

[4] 薛定宇. 控制系统计算机辅助设计 —— MATLAB 语言与应用 (第 2 版) [M]. 北京: 清华大学出版社, 2006

[5] Frederick D K, Rimer M. Benchmark problem for CACSD packages [A]. Abstracts of the second IEEE symposium on computer-aided control system design [C]. Santa Barbara, USA, 1985

第 3 章 线性代数问题的计算机求解

线性代数问题是科学技术中最常见的数学问题，很多理论和应用都是建立在线性代数的基础上的，因此解决线性代数问题是很有意义的。然而经典线性代数的课程中介绍的都是手工推导的方法，不适合于高阶矩阵的分析与计算，所以需要计算机数学语言来解决这些高阶问题。

很多计算机数学语言，如 MATLAB 语言，都起源于对线性代数问题的研究。早期的线性代数计算问题侧重于数值解法[1,2]，很多数学软件包也都是从线性代数的计算开始的。例如，国际上最著名的 EISPACK 是求解矩阵特征值问题的软件包，LINPACK 是求解一般线性代数问题的软件包，目前最新的 LaPACK 也是解决线性代数计算的软件包。随着计算机科学的发展，当前能解决矩阵分析与运算问题的计算机数学语言已经不局限于数值线性代数方法了，逐渐也可以求解解析解问题。Mathmetica 和 Maple 等大型计算机数学语言都已经能直接求解线性代数的解析解问题。MATLAB 语言的符号运算工具箱可以调用 Maple 的各种解析运算功能，可以很好地解决线性代数的解析解运算问题。

线性代数是线性系统中最重要的理论基础。基于线性代数理论，建立了在控制理论中有重要影响的状态空间模型和状态空间理论。本章首先在 3.1 节中介绍如何在 MATLAB 中输入各种特殊矩阵的方法，然后介绍线性系统的 MATLAB 表示方法。3.2 节将系统介绍矩阵分析的理论与求解，首先介绍矩阵的行列式、秩、范数等概念与求解方法，然后介绍矩阵求逆、特征值特征向量求取等内容。由矩阵分析理论可以容易地进行线性系统中最重要的一些性质的分析，如系统的稳定性、可控性和可观测性等，可以利用矩阵逆矩阵和求逆的方法建立起状态方程到传递函数模型的转换方法以及系统的频域分析方法等，可以进行状态反馈的极点配置设计，还可以利用特征值估算的 Gershgorin 定理建立起多变量系统的频域分析体系。3.3 节将介绍矩阵的基本变换与分解方法，首先给出矩阵相似变换及正交矩阵的基本

概念，然后介绍矩阵的三角分解和 Cholesky 分解，给出正定矩阵的概念，并介绍几种常用的矩阵变换方法，如伴随矩阵变换、Jordan 矩阵变换以及矩阵的奇异值分解等。基于矩阵分解和线性变换方法，可以很容易地建立起线性系统的相似变换理论与方法，对一般的线性系统模型变换出有用的可控性、可观测性、Jordan 及 Luenberger 标准型，并可以对系统进行结构分解，获得系统的最小实现模型。还可以根据正定矩阵和正定函数的概念，介绍利用 Lyapunov 直接方法对非线性系统稳定性进行定性分析。3.4 节将介绍各种矩阵方程的求解方法，如一般的线性代数方程，一般意义下的 Lyapunov 方程和 Sylvester 方程的求解，还将介绍 Riccati 二次型方程的求解方法，基于矩阵方程，介绍控制系统的 Gram 矩阵求解方法、系统范数指标的定义与求解、线性二次型最优控制等线性系统中重要的分析与设计方法。3.5 节将介绍矩阵函数的求解方法，以常用的指数函数和三角函数为例介绍矩阵函数的解析解方法，还给出了矩阵一般函数的解析解方法。基于此理论介绍了控制系统连续和离散模型的相互转换方法，连续状态方程解析解方法等，为线性系统的分析和设计打下了基础。

3.1 特殊矩阵的输入

前面介绍了一般实数或复数矩阵的 MATLAB 输入方法。在实际应用中，经常存在一些特殊的矩阵，例如单位矩阵，由于其特殊性无须用底层方法去输入，所以本节介绍一些特殊矩阵的输入方法，并将引入特殊矩阵的符号表示方法。

3.1.1 数值矩阵的输入

1. 零矩阵、幺矩阵及单位矩阵

在一般的矩阵理论中，把所有元素都为零的矩阵定义成零矩阵，把元素全为 1 的矩阵称为幺矩阵，把主对角线元素均为 1，而其他元素全部为 0 的方阵称为单位矩阵。这里进一步扩展单位矩阵的定义，使其为 $m \times n$ 的矩阵。零矩阵、幺矩阵和扩展单位矩阵的 MATLAB 生成函数分别为

```
A=zeros(n),  B=ones(n),  C=eye(n)      % 生成 n×n 方阵
A=zeros(m,n); B=ones(m,n); C=eye(m,n)  % 生成 m×n 矩阵
A=zeros(size(B))     % 生成和矩阵 B 同样维数的矩阵
```

例 3-1 下面的语句可以生成一个 3×8 的零矩阵 A，并可以生成一个和 A 维数相同的扩展单位阵 B。可见，这些特殊矩阵的输入还是很容易的。

```
>> A=zeros(3,8),     % 零矩阵输入
   B=eye(size(A))    % 单位矩阵输入
```

前面两条语句将分别给下面两个矩阵赋值

$$\boldsymbol{A}=\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\end{bmatrix}$$

函数 `zeros()` 和 `ones()` 还可用于多维数组的生成，例如，`zeros(3,4,5)` 将生成一个 $3\times4\times5$ 的三维数组，其元素全部为 0。

2. 对角元素矩阵

对角矩阵是一种特殊的矩阵，这种矩阵的主对角线元素可以为 0 或非 0 元素，而非对角线元素的值均为 0。对角矩阵的数学描述方法为 $\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)$，其中对角矩阵的数学表示为

$$\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)=\begin{bmatrix}\alpha_1&&&\\&\alpha_2&&\\&&\ddots&\\&&&\alpha_n\end{bmatrix} \tag{3-1-1}$$

MATLAB 提供了对角矩阵的生成函数 `diag()`。该函数的调用格式为

```
A=diag(V)       % 已知向量生成对角矩阵
V=diag(A)       % 已知矩阵提取对角元素列向量
A=diag(V,k)     % 生成主对角线上第 k 条对角线为 V 的矩阵
```

例 3-2 MATLAB 中的 `diag()` 函数是很有特色的，其不同方式执行不同的任务。例如，直接使用此语句可以分别输入各种矩阵

```
>> C=[1 2 3]; V=diag(C)          % 由给定向量构造对角矩阵
   V1=diag(V)                    % 由矩阵提取对角元素得出行向量
   C=[1 2 3]; V2=diag(C,2)       % 由次对角线元素构造矩阵
   V3=diag([1 2 3 4])+diag([2 3 4],1)+diag([5 4 3],-1)
```

这样，前面的语句将给下面矩阵赋值

$$\boldsymbol{V}=\begin{bmatrix}1&0&0\\0&2&0\\0&0&3\end{bmatrix},\quad \boldsymbol{V}_1=\begin{bmatrix}1\\2\\3\end{bmatrix},\quad \boldsymbol{V}_2=\begin{bmatrix}0&0&1&0&0\\0&0&0&2&0\\0&0&0&0&3\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},\quad \boldsymbol{V}_3=\begin{bmatrix}1&2&0&0\\5&2&3&0\\0&4&3&4\\0&0&3&4\end{bmatrix}$$

在实际应用中还可以取 k 为负值，表示主对角线下数的第 k 条对角线。利用这样的性质，由上面语句构造出了三对角矩阵 $\boldsymbol{V}_3$。

如果有若干个子矩阵 $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_n$，可以编写一个 diagm() 函数，构造块对角矩阵。该函数的清单为

```
function A=diagm(varargin)
A=[];
for i=1:length(varargin), B=varargin{i};
   [nA,mA]=size(A); [nB,mB]=size(B); A(nA+1:nA+nB,mA+1:mA+mB)=B;
end
```

该函数的调用格式为 $\boldsymbol{A}$=diagm($\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_n$)，其中，子矩阵个数是任意多的。该函数可以得出块对角矩阵

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_1 & & & \\ & \boldsymbol{A}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{A}_n\end{bmatrix} \tag{3-1-2}$$

3. Hankel 矩阵

Hankel 矩阵的一般形式如下：

$$\boldsymbol{H}=\begin{bmatrix}c_1 & c_2 & \cdots & c_m \\ c_2 & c_3 & \cdots & c_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+m-1}\end{bmatrix} \tag{3-1-3}$$

如果 $n \to \infty$，则可以构造无穷型 Hankel 矩阵。Hankel 矩阵是对称矩阵，且其反对角线上所有的元素都相同。

在 MATLAB 语言中，给定两个向量 $\boldsymbol{c}$ 和 $\boldsymbol{r}$，如果用 $\boldsymbol{H}$=hankel($\boldsymbol{c}$,$\boldsymbol{r}$)，来生成 $\boldsymbol{H}$，则首先将 $\boldsymbol{H}$ 矩阵第一列的各个元素定义为 $\boldsymbol{c}$ 向量，将最后一行各个元素定义为 $\boldsymbol{r}$，这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵。根据 Hankel 矩阵的性质，其最后一行的第一个元素应该等于第 1 列的最后一个元素，如果冲突将给出元素冲突的警告信息。

如果已知一个向量 $\boldsymbol{c}$，则也可以由 $\boldsymbol{H}$=hankel($\boldsymbol{c}$) 函数来构造出一个 Hankel 矩阵。将 $\boldsymbol{H}$ 矩阵的第一列的各个元素定义为 $\boldsymbol{c}$ 向量，这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵，使得下三角矩阵均为 0。

例 3-3 试用 MATLAB 语句输入下面给出的 Hankel 矩阵 $\boldsymbol{H}$。

$$\boldsymbol{H}=\begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9\end{bmatrix}$$

求解 分析给出的矩阵，用向量分别表示该矩阵的首列和最后一行，则可以由下面语句生成 Hankel 矩阵。如果只给出一个向量 C，则可以生成下三角矩阵为 0 的方阵。

```
>> C=[1 2 3]; R=[3 4 5 6 7 8 9]; H=hankel(C,R)
   C=[1 2 3]; H1=hankel(C)
```

这样输入的两个矩阵分别为

$$\boldsymbol{H}=\begin{bmatrix}1&2&3&4&5&6&7\\2&3&4&5&6&7&8\\3&4&5&6&7&8&9\end{bmatrix},\ \boldsymbol{H}_1=\begin{bmatrix}1&2&3\\2&3&0\\3&0&0\end{bmatrix}$$

4. Hilbert 矩阵及逆 Hilbert 矩阵

Hilbert 矩阵是一类特殊矩阵，它的第 (i,j) 元素的值满足 $h_{i,j}=1/(i+j-1)$，这时一个 $n\times n$ 阶的 Hilbert 矩阵可以写成

$$\boldsymbol{H}=\begin{bmatrix}1&1/2&1/3&\cdots&1/n\\1/2&1/3&1/4&\cdots&1/(n+1)\\\vdots&\vdots&\vdots&\ddots&\vdots\\1/n&1/(n+1)&1/(n+2)&\cdots&1/(2n-1)\end{bmatrix}\tag{3-1-4}$$

产生 Hilbert 矩阵的 MATLAB 函数为 $\boldsymbol{A}$=`hilb(`n`)`，其中，n 为要产生的矩阵阶次。

高阶 Hilbert 矩阵一般为坏条件的矩阵，所以直接对之求逆往往会引出浮点溢出的现象。MATLAB 提供了直接求取逆 Hilbert 矩阵的算法及函数，其调用格式为 $\boldsymbol{B}$=`invhilb(`n`)`。

由于 Hilbert 矩阵本身接近奇异的性质，所以在处理该矩阵时建议尽量采用符号运算工具箱，而采用数值解时应该检验结果的正确性。

5. Vandermonde 矩阵

假设有一个序列 $\boldsymbol{c}$，其各个元素为 $\{c_1,c_2,\cdots,c_n\}$，则可以写出一个矩阵，其第 (i,j) 元素满足 $v_{i,j}=c_i^{n-j}$, $i,j=1,2,\cdots,n$。这样可以构成一个矩阵

$$\boldsymbol{V}=\begin{bmatrix}c_1^{n-1}&c_1^{n-2}&\cdots&c_1&1\\c_2^{n-1}&c_2^{n-2}&\cdots&c_2&1\\\vdots&\vdots&\ddots&\vdots&\vdots\\c_n^{n-1}&c_n^{n-2}&\cdots&c_n&1\end{bmatrix}\tag{3-1-5}$$

该矩阵称作 Vandermonde 矩阵。如果已知一个向量 $\boldsymbol{c}$，则可以由 MATLAB 提供的 `vander()` 函数来构造一个 Vandermonde 矩阵 $\boldsymbol{V}$=`vander(`$\boldsymbol{c}$`)`。

例 3-4 若向量 $\boldsymbol{C}$=`[1,2,3,4,5]`，则可以得出该向量对应的 Vandermonde 矩阵为

```
>> C=[1, 2, 3, 4, 5];  V=vander(C)
```

这样构造出的 Vandermonde 矩阵为 $\boldsymbol{V}=\begin{bmatrix}1&1&1&1&1\\16&8&4&2&1\\81&27&9&3&1\\256&64&16&4&1\\625&125&25&5&1\end{bmatrix}$。

6. 伴随矩阵

假设有一个首一化的多项式

$$P(s)=s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n \tag{3-1-6}$$

则可以写出一个伴随矩阵

$$\boldsymbol{A}_\mathrm{c}=\begin{bmatrix}-a_1&-a_2&\cdots&-a_{n-1}&-a_n\\1&0&\cdots&0&0\\0&1&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\0&0&\cdots&1&0\end{bmatrix} \tag{3-1-7}$$

生成伴随矩阵的 MATLAB 函数调用格式为 **B=compan(p)**，其中，$\boldsymbol{p}$ 为一个多项式的系数向量，该函数将自动对多项式进行首一化处理。

例 3-5 考虑一个多项式 $P(s)=2s^4+4s^2+5s+6$，试写出该多项式的伴随矩阵。

求解 先输入特征多项式，则伴随矩阵可以通过下面的语句建立起来，赋给 $\boldsymbol{A}$ 矩阵。

```
>> P=[2 0 4 5 6]; A=compan(P)
```

则可以得出矩阵 $\boldsymbol{A}=\begin{bmatrix}0&-2&-2.5&-3\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}$。

7. 随机元素矩阵

顾名思义，随机元素矩阵的各个元素是随机产生的。如果矩阵的随机元素满足 $[0,1]$ 区间上的均匀分布，则可以由 MATLAB 函数 rand() 来生成，该函数通常的调用格式为

```
A=rand(n)      % 生成 n×n 阶标准均匀分布伪随机数方阵
A=rand(n,m)    % 生成 n×m 阶标准均匀分布伪随机数矩阵
```

函数 rand() 还可以生成多维数组。满足标准正态分布的随机数矩阵可以由 randn() 函数获得，当然也可以使用 **B=rand(size(A))** 形式调用该函数。

这里的随机数实际上是“伪随机数”。所谓伪随机数，就是通过某种数学公式生成的、满足某些随机指标的数据。这样的随机数是可以重复的，与某些用电子方法获得的不可重复的随机数是不同的。

更一般地，如果想生成在 (a,b) 区间上均匀分布的随机数矩阵，则可以首先用 $\boldsymbol{V}$=rand(n,m) 命令生成一个在 $(0,1)$ 区间上的均匀分布随机数矩阵 $\boldsymbol{V}$，再用 $\boldsymbol{V}_1=a+(b-a)*\boldsymbol{V}$ 语句则可以生成满足需要的矩阵 $\boldsymbol{V}_1$。如果想生成满足 $N(\mu,\sigma^2)$ 的正态分布的随机数，则可以先用 $\boldsymbol{V}$=randn(n,m) 命令生成标准随机分布的随机数矩阵 $\boldsymbol{V}$，再用 $\boldsymbol{V}_1=\mu+\sigma*\boldsymbol{V}$ 命令就可以转换成所需的矩阵。

MATLAB 的统计工具箱还提供了大量满足特殊分布的随机数生成函数，如满足 Rayleigh 分布的 raylrnd()，满足 χ^2 分布的 chi2rnd()，具体请参见文献 [3,4]。

3.1.2 符号矩阵的输入

如果已经建立起了数值矩阵 $\boldsymbol{A}$，则可以由 $\boldsymbol{B}$=sym($\boldsymbol{A}$) 语句将其转换成符号矩阵。这样，所有数值矩阵均可以通过这样的形式转换成符号矩阵，并利用符号运算工具箱获得更高精度的解。

对于一些特殊矩阵形式，如 Vandermonde 矩阵、Hankel 矩阵及伴随矩阵，符号运算工具箱不直接支持它们，所以需要编写下面的一些函数，将其置于 @sym 目录下，这样才可以对符号向量建立起这些符号矩阵。

参考 MATLAB 语言对数字矩阵生成的相应函数，可以改写出适合符号运算的新函数。例如，可以编写出生成伴随矩阵的 MATLAB 函数 compan()。

```
function A=compan(c)
c=c(:).'; A=sym(diag(ones(1,n-2),-1)); A(1,:)=-c(2:n)./c(1);
```

例 3-6 试用解析方法建立起下面多项式的伴随矩阵。

$$P(\lambda)=a_1\lambda^8+a_2\lambda^7+a_3\lambda^6+\cdots+a_7\lambda^2+a_8\lambda+a_9$$

求解 由上面编写的函数，可以先申明符号变量，并以向量形式输入多项式，最后通过下面的语句直接建立所需的伴随矩阵。

```
>> syms a1 a2 a3 a4 a5 a6 a7 a8 a9
   A=compan([a1 a2 a3 a4 a5 a6 a7 a8 a9])
```

这样建立起所需的矩阵

$$\boldsymbol{A}=\begin{bmatrix} -a_2/a_1 & -a_3/a_1 & -a_4/a_1 & -a_5/a_1 & -a_6/a_1 & -a_7/a_1 & -a_8/a_1 & -a_9/a_1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

类似地，Hankel 矩阵的 MATLAB 函数 hankel() 为

```
function H=hankel(c,r)
c=c(:); nc=length(c);
if nargin==1, r=zeros(size(c)); end
r=r(:); nr=length(r); x=[c; r((2:nr)')]; cidx=(1:nc)';
ridx=0:(nr-1); H1=cidx(:,ones(nr,1))+ridx(ones(nc,1),:); H=x(H1);
```

还可以建立起生成 Vandermonde 矩阵的 MATLAB 函数 vander()，其格式与 MATLAB 语言中的数值函数完全一致。

```
function A=vander(v)
n=length(v); v=v(:); A=sym(ones(n));
for j=n-1:-1:1, A(:,j)=v.*A(:,j+1); end
```

3.1.3 线性系统的状态空间模型

控制系统的状态空间描述和前面介绍的传递函数描述是刻画同一问题的两种不同方法。传递函数模型描述的是系统的输入和输出之间的关系，而状态方程模型除此之外，还描述了一些内部的状态变量，所以状态方程模型又称为系统的内部模型，而传递函数模型称为系统的外部模型。系统的状态方程模型构造了“现代控制理论”的基础。

这里首先介绍系统的状态方程描述方法，然后着重介绍连续和离散线性时不变系统的数学模型及其 MATLAB 表示方法。

1. 连续系统的状态方程模型

在介绍系统状态方程之前先给出一个实际系统状态方程建模的例子，然后给出线性系统状态方程的一般形式。

例 3-7 重新考虑图 2-3 中的 RLC 串联电路，式 (2-2-10) 表示出该系统的微分方程模型。试选择状态变量，推导出对应的状态方程模型。如果输入信号 $u(t)$ 为阶跃信号，试求出 $u_{\rm c}(t)$ 的解析解。

求解 选择状态变量 $x_1(t)=i(t)$, $x_2=u_{\rm c}(t)$，由式 (2-2-8) 可以得出 $x_1=C\dot{x}_2$，由式 (2-2-9) 可以得出 $u(t)=Rx_1+L\dot{x}_1+x_2$，从而可以写出系统的状态方程为

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-R/L & -1/L\\ 1/C & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1/L\\ 0\end{bmatrix}u(t),\ y=[0,1]\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

若选择状态变量 $x_1=u_{\rm c}$, $x_2=i$，则可以写出 $x_2=C\dot{x}_1$, $u(t)=Rx_2+L\dot{x}_2+x_1$，这样可以改写状态方程为

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1/C\\ -1/L & -R/L\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0 & 1/L\end{bmatrix}u(t),\ \ y=[1,0]\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

可见，随着状态变量的不同选择，得出的状态方程模型是不惟一的。

连续线性时不变系统状态方程的一般表示为

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases} \tag{3-1-8}$$

式中 $\boldsymbol{u} = [u_1, \cdots, u_p]^{\mathrm{T}}$ 与 $\boldsymbol{y} = [y_1, \cdots, y_q]^{\mathrm{T}}$ 分别为系统的输入和输出向量，$\boldsymbol{x}$ 为系统的状态向量。矩阵 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 和 $\boldsymbol{D}$ 为维数相容的矩阵。这里维数相容是指在方程里相应的项是可乘的。准确地说，$\boldsymbol{A}$ 矩阵是 $n \times n$ 方阵，$\boldsymbol{B}$ 为 $n \times p$ 矩阵，$\boldsymbol{C}$ 为 $q \times n$ 矩阵，$\boldsymbol{D}$ 为 $q \times p$ 矩阵。

在 MATLAB 中表示系统的状态方程模型是相当直观的，只需要将各个系数矩阵按照常规矩阵的方式输入到工作空间中即可，这样，系统的状态方程模型可以用下面的语句直接建立起来 *G*=ss(***A***,***B***,***C***,***D***) 。

例 3-8 多变量系统的状态方程模型可以用前面介绍的方法直接输入，无需再进行特殊的处理。试将如下双输入双输出系统的状态方程模型输入 MATLAB 环境。

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -12 & -17.2 & -16.8 & -11.9 \\ 6 & 8.6 & 8.4 & 6 \\ 6 & 8.7 & 8.4 & 6 \\ -5.9 & -8.6 & -8.3 & -6 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1.5 & 0.2 \\ 1 & 0.3 \\ 2 & 1 \\ 0 & 0.5 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

求解 系统的状态方程模型可以用下面的语句直接输入

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5]; C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D) % 输入并显示系统状态方程模型
```

2. 离散系统的状态方程模型

离散系统状态方程模型可以表示为

$$\begin{cases} \boldsymbol{x}[(k+1)T] = \boldsymbol{F}\boldsymbol{x}(kT) + \boldsymbol{G}\boldsymbol{u}(kT) \\ \boldsymbol{y}(kT) = \boldsymbol{C}\boldsymbol{x}(kT) + \boldsymbol{D}\boldsymbol{u}(kT) \end{cases} \tag{3-1-9}$$

这里 T 为离散系统的采样周期。可以看出，该模型的输入应该与连续系统状态方程一样，只需输入 $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{C}$ 和 $\boldsymbol{D}$ 矩阵，就可以用 ss() 函数将该模型输入到 MATLAB 的工作空间了：*H*=ss(***F***,***G***,***C***,***D***,'Ts',*T*) 。

3.2 矩阵基本分析

3.2.1 矩阵基本概念与性质

1. 矩阵的行列式

矩阵 $\boldsymbol{A}=\{a_{ij}\}$ 的行列式定义为

$$\boldsymbol{D}=|\ \boldsymbol{A}\ |=\det(\boldsymbol{A})=\sum(-1)^k a_{1k_1}a_{2k_2}\cdots a_{nk_n} \tag{3-2-1}$$

式中，$k_1,k_2,\cdots,k_n$ 是将序列 $1,2,\cdots,n$ 的元素交换 k 次所得出的一个序列，每个这样的序列称为一个置换 (permutation)；而 Σ 表示对 $k_1,k_2,\cdots,k_n$ 取遍 $1,2,\cdots,n$ 的所有排列的求和。

计算矩阵的行列式有多种算法，在 MATLAB 中采用的方法是对原矩阵 $\boldsymbol{A}$ 进行三角分解(又称为 LU 分解，后面将介绍)，将其分解成一个上三角矩阵 $\boldsymbol{U}$ 和一个下三角矩阵 $\boldsymbol{L}$ 的积，即 $\boldsymbol{A}=\boldsymbol{L}\boldsymbol{U}$，这样可以先求出 $\boldsymbol{L}$ 矩阵的行列式。注意，在这一矩阵中只有一种非 0 的排列方式且其行列式的值 s 为 1 或 -1。同样，因为 $\boldsymbol{U}$ 为上三角矩阵，所以其行列式的值为该矩阵主对角线元素之积，即 $\boldsymbol{A}$ 矩阵行列式为 $\det(\boldsymbol{A})=s\prod_{i=1}^n u_{ii}$。MATLAB 提供了内在函数 `det()`，其调用格式很直观，为 `d=det(A)`，利用它可以直接求取矩阵 $\boldsymbol{A}$ 的行列式。该函数同样适用于符号矩阵 $\boldsymbol{A}$。

例 3-9 试求出矩阵 $\boldsymbol{A}=\begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix}$ 的行列式。

求解 由下面的语句可以立即得出矩阵的行列式

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];  det(A)
```

该矩阵的行列式为 0，表明矩阵 $\boldsymbol{A}$ 奇异。

例 3-10 高阶 Hilbert 矩阵是接近奇异的矩阵。试用解析解方法计算出 20×20 的 Hilbert 矩阵的行列式。

求解 首先用 `hilb()` 函数可以定义一个 20×20 的 Hilbert 矩阵，将其转换成符号矩阵，则 MATLAB 的 `det()` 函数会自动采用解析解法求出其行列式的值。

```
>> tic, A=sym(hilb(20)); det(A), toc
```

可以得出如下行列式的解析解及近似值为

$$\det(\boldsymbol{H})=\frac{1}{\underbrace{2377454716\cdots 36800000000000000000000000000000000000}_{225\text{ 位，因排版限制省略了中间的数字}}}\approx 4.2\times 10^{-224}$$

并得出运行时间为 0.862 秒。从计算结果还可以看出，利用解析方法在 1 秒内就可以

得出原问题的解析解，因为这里采用的方法是矩阵三角分解的方法，而不是代数余子式算法。

2. 矩阵的迹

假设一个方阵为 $\boldsymbol{A}=\{a_{ij}\}$, $i,j=1,2,\cdots,n$，则矩阵 $\boldsymbol{A}$ 的迹定义为该矩阵对角线上各个元素之和

$$\mathrm{tr}(\boldsymbol{A})=\sum_{i=1}^{n}a_{ii} \tag{3-2-2}$$

由代数理论可知，矩阵的迹和该矩阵的特征值之和是相同的，矩阵 $\boldsymbol{A}$ 的迹可以由 MATLAB 函数 trace() 求出，该函数的调用和数学表示相似 t=trace($\boldsymbol{A}$)。其实，矩阵的迹还可以由底层命令 t=sum(diag($\boldsymbol{A}$)) 求出。

例 3-9 中矩阵的迹可以由 MATLAB 语句直接求出：trace($\boldsymbol{A}$)=34。

3. 矩阵的秩

若矩阵所有的列向量中共有 r_{c} 个线性无关，则称矩阵的列秩为 r_{c}。如果 $r_{\mathrm{c}}=m$，则称 $\boldsymbol{A}$ 为列满秩矩阵。相应地，若矩阵 $\boldsymbol{A}$ 的行向量中有 r_{r} 个是线性无关的，则称矩阵 $\boldsymbol{A}$ 的行秩为 r_{r}。如果 $r_{\mathrm{r}}=n$，则称 $\boldsymbol{A}$ 为行满秩矩阵。可以证明，矩阵的行秩和列秩是相等的，故称之为矩阵的秩，记作

$$\mathrm{rank}(\boldsymbol{A})=r_{\mathrm{c}}=r_{\mathrm{r}} \tag{3-2-3}$$

这时，矩阵的秩为 rank($\boldsymbol{A}$)。矩阵的秩也表示该矩阵中行列式不等于 0 的子式的最大阶次。所谓子式，即为从原矩阵中任取 k 行及 k 列所构成的子矩阵。

矩阵求秩的算法也是多种多样的，其区别是有的算法是稳定的，而有的算法可能因矩阵的条件数变化不是很稳定。MATLAB 中采用的算法是基于矩阵的奇异值分解的算法[5]。首先对矩阵进行奇异值分解，得出矩阵 $\boldsymbol{A}$ 的 n 个奇异值 σ_i, $i=1,2,\cdots,n$，在这 n 个奇异值中找出大于给定误差限 ε 的个数 r，这时 r 就可以认为是 $\boldsymbol{A}$ 矩阵的秩。

MATLAB 提供了一个内在函数 rank()，用它可以求取给定矩阵的秩。该函数的调用格式为

```
r=rank(A)      % 用默认的精度求数值秩
r=rank(A,ε)    % 给定精度 ε 下求数值秩
```

其中，$\boldsymbol{A}$ 为给定矩阵，ε 为机器精度。符号运算工具箱中也提供了 rank() 函数，可以求出数值矩阵秩的解析解，其调用格式与前面的方法完全一致。

例 3-11 试求出例 3-9 中给出的 $\boldsymbol{A}$ 矩阵的秩。

求解 用 rank($\boldsymbol{A}$) 函数可以得出该矩阵的秩为

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];  rank(A)
```

该矩阵的秩为 3，小于矩阵的阶次，故矩阵 $\boldsymbol{A}$ 是非满秩矩阵。

例 3-12 现在考虑例 3-10 中给出的 20×20 Hilbert 矩阵，考虑用数值方法和解析方法分别求该矩阵的秩，并比较其正确性。

求解 先考虑数值方法，应该给出命令

```
>> H=hilb(20); rank(H)
```

故而可以得出结论。因为该矩阵的秩为 13，该值和矩阵阶次相差太多，所以 $\boldsymbol{H}$ 矩阵为非满秩矩阵。其实该函数对一些接近奇异的矩阵可能出现错误结论，用数值解的方法应该注意。如果有可能应该采用解析解的方法求解该问题，该方法将得出 $\boldsymbol{H}$ 矩阵的秩为 20，为满秩矩阵。

```
>> H=sym(hilb(20)); rank(H)   % 可见原矩阵为非奇异矩阵
```

4. 矩阵范数

矩阵的范数是对矩阵的一种测度。在介绍矩阵的范数之前，首先要介绍向量范数的基本概念。如果对线性空间中的一个向量 $\boldsymbol{x}$ 存在一个函数 $\rho(\boldsymbol{x})$ 满足下面 3 个条件：

① $\rho(\boldsymbol{x})\geqslant 0$ 且 $\rho(\boldsymbol{x})=0$ 的充要条件是 $\boldsymbol{x}=\boldsymbol{0}$

② $\rho(a\boldsymbol{x})=|a|\rho(\boldsymbol{x})$, a 为任意标量

③ 对向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 有 $\rho(\boldsymbol{x}+\boldsymbol{y})\leqslant\rho(\boldsymbol{x})+\rho(\boldsymbol{y})$

则称 $\rho(\boldsymbol{x})$ 为 $\boldsymbol{x}$ 向量的范数。范数的形式是多种多样的。可以证明，下面给出的一族式子都满足上述的 3 个条件。

$$||\boldsymbol{x}||_p=\left(\sum_{i=1}^{n}|\ x_i\ |^p\right)^{1/p},\ p=1,2,\cdots,\text{且}\ ||\boldsymbol{x}||_\infty=\max_{1\leqslant i\leqslant n}|\ x_i\ | \tag{3-2-4}$$

这里用到了向量范数的记号 $||\boldsymbol{x}||_p$。向量范数可以由 MATLAB 底层命令直接解出 `sum(abs(A).^p)^(1/p)` 。

矩阵的范数定义比向量的稍复杂一些，其数学定义为：对于任意的非零向量 $\boldsymbol{x}$，矩阵 $\boldsymbol{A}$ 的范数为

$$||\boldsymbol{A}||=\sup_{\boldsymbol{x}\neq 0}\frac{||\boldsymbol{A}\boldsymbol{x}||}{||\boldsymbol{x}||} \tag{3-2-5}$$

和向量的范数一样，对矩阵来说也有常用的范数定义方法

$$||\boldsymbol{A}||_1=\max_{1\leqslant j\leqslant n}\sum_{i=1}^{n}|\ a_{ij}\ |,\ ||\boldsymbol{A}||_2=\sqrt{s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})},\ ||\boldsymbol{A}||_\infty=\max_{1\leqslant i\leqslant n}\sum_{j=1}^{n}|\ a_{ij}\ | \tag{3-2-6}$$

其中，$s(\boldsymbol{X})$ 为 $\boldsymbol{X}$ 矩阵的特征值，而 $s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$ 为 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 矩阵的最大特征值。事实上，$||\boldsymbol{A}||_2$ 还等于 $\boldsymbol{A}$ 矩阵的最大奇异值。

MATLAB 提供了求取矩阵范数的函数 norm()，允许求各种意义下的矩阵范数。该函数的调用格式为

```
N=norm(A)          % 求解默认的 ||A||2
N=norm(A, 选项)    % 选项可以为 1,2,inf,'fro' 等
```

这样，例 3-9 中矩阵 $\boldsymbol{A}$ 的各种范数可以由下面的 MATLAB 函数直接求出。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   [norm(A),norm(A,2),norm(A,1),norm(A,Inf),norm(A,'fro')]
```

可以得出 $||\boldsymbol{A}||_1 = ||\boldsymbol{A}||_2 = ||\boldsymbol{A}||_\infty = 34$，$||\boldsymbol{A}||_{\rm F} = 38.6782$。这里有两点值得注意，首先 norm($\boldsymbol{A}$) 和 norm($\boldsymbol{A}$,2) 应该给出同样的结果，因为它们都表示 $||\boldsymbol{A}||_2$，其次因为巧合，在这个例子中，$||\boldsymbol{A}||_1 = ||\boldsymbol{A}||_\infty$。但一般情况下，$||\boldsymbol{A}||_1 = ||\boldsymbol{A}||_\infty$ 不一定能满足。

遗憾的是，符号运算工具箱中未提供 norm() 函数。故若需要求解数值矩阵的范数，应该先将矩阵用 double() 函数转换成双精度数值矩阵，然后再调用数值矩阵的 norm() 函数。

5. 特征多项式

引入算子 s，并构造一个矩阵 $s\boldsymbol{I}-\boldsymbol{A}$，再求出该矩阵的行列式，则可以得出一个关于算子 s 的多项式

$$C(s) = \det(s\boldsymbol{I}-\boldsymbol{A}) = s^n + c_2 s^{n-1} + \cdots + c_n s + c_{n+1} \tag{3-2-7}$$

这样的多项式 $C(s)$ 称为矩阵 $\boldsymbol{A}$ 的特征多项式。其中，系数 c_i，$i=2,3,\cdots,n+1$ 称为矩阵的特征多项式系数。多项式首项系数 $c_1=1$。

MATLAB 提供了求取矩阵特征多项式系数的函数 poly()，该函数的调用格式为 $\boldsymbol{c}$=poly($\boldsymbol{A}$)，返回的 $\boldsymbol{c}$ 为一个行向量，其各个分量为矩阵 $\boldsymbol{A}$ 的降幂排列的特征多项式系数。该函数的另外一种调用格式是，如果给定的 $\boldsymbol{A}$ 为向量，则假定该向量是一个矩阵的特征值，由此求出该矩阵的特征多项式系数，如果向量 $\boldsymbol{A}$ 中有无穷大或 NaN 值，则首先剔除它，再计算特征多项式的系数。

需要指出的是，如果 $\boldsymbol{A}$ 为符号矩阵，该函数仍然适用，但得出的不是系数向量，而是多项式的数学表达式本身。

例 3-13 试求出例 3-9 中给出的 $\boldsymbol{A}$ 矩阵的特征多项式。

求解 可以通过下面的 poly() 函数直接求出该矩阵的特征多项式。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; p=poly(A)
   p0=[1 -34 -80 2720 0]; norm(p-p0)
```

得出特征多项式系数为 $1, -34, -80, 2720, -1.813\times10^{-13}$。该系数和理论值的差的范数为 2.2810×10^{-12}。

用符号运算工具箱中的 poly() 函数同样可以求出矩阵的特征多项式。

```
>> A=sym(A); poly(A)
```

由解析解方法可以得出其特征多项式为 $x^4 - 34x^3 - 80x^2 + 2720x$。

在实际应用中还有其他简单的数值方法可以精确地求出矩阵的特征多项式系数。例如，下面给出的 Leverrier-Faddeev 递推算法是求取矩阵的特征多项式的一种有效的递推算法。

$$c_{k+1} = -\frac{1}{k}\mathrm{tr}(\boldsymbol{A}\boldsymbol{R}_k),\ \ \boldsymbol{R}_{k+1} = \boldsymbol{A}\boldsymbol{R}_k + c_{k+1}\boldsymbol{I},\ \ k = 1, \cdots, n \tag{3-2-8}$$

其中 $\boldsymbol{R}_1 = \boldsymbol{I}, c_1 = 1$。该算法首先给出一个单位阵 $\boldsymbol{I}$，并将之赋给 $\boldsymbol{R}_1$，然后对每个 k 的值分别求出特征多项式参数，并更新 $\boldsymbol{R}_k$ 矩阵，最终得出矩阵的特征多项式系数 c_k。该算法可以直接由下面的 MATLAB 语句编写一个 poly1() 函数实现，该函数还实现了 poly() 函数的其他调用方法，可以全面取代该函数。

```
function c=poly1(A)
[nr,nc]=size(A);
if nc==nr % 给出若为方阵，则用 Leverrier-Faddeev 算法求特征多项式
   I=eye(nc);  R=I; c=[1 zeros(1,nc)];
   for k=1:nc, c(k+1)=-1/k*trace(A*R); R=A*R+c(k+1)*I; end
elseif (nr==1 | nc==1) % 给出为向量时，构造矩阵
   A=A(isfinite(A)); n=length(A); c=[1 zeros(1,n)];
   for j=1:n, c(2:(j+1))=c(2:(j+1))-A(j).*c(1:j); end
else   % 参数有误则给出错误信息
   error('Argument must be a vector or a square matrix.')
end
```

调用新的 poly1($\boldsymbol{A}$) 函数，则可以得出精确结果。

例 3-14 试求向量 $\boldsymbol{B} = [a_1, a_2, a_3, a_4, a_5]$ 对应 Vandermonde 矩阵的特征多项式。

求解 可以用 $\boldsymbol{A}$=vander($\boldsymbol{B}$) 函数构造一个 Vandermonde 矩阵 $\boldsymbol{A}$，这样就能直接使用 poly($\boldsymbol{A}$) 函数获得该矩阵的特征多项式。

```
>> syms a1 a2 a3 a4 a5 x; A=vander([a1 a2 a3 a4 a5]);
   collect(poly(A),x) % 按 x 合并同类项，化简多项式
```

该矩阵的特征多项式数学表示为

$$\begin{aligned}\det(\boldsymbol{A}) = & x^5 + (-a_3 - a_1 - a_5)x^4 + (a_5a_1 + a_3a_1 + a_5a_3 - 2a_4^2 - 2a_5^2 - a_2^2 - a_3^2)x^3 \\ & + (a_4^2a_1 + a_5^2a_3 + a_4^2a_5 + a_2^2a_5 + a_4^2a_3 - a_3a_1a_5 - 2a_2a_5a_4 + a_5^2a_1 - 2a_2a_4a_3 + 2a_5^3 + a_3^3)x^2 \\ & + (-3a_4^2a_3a_5 + a_5^4 + a_4^2a_5^2 + a_4^4 - a_5^3a_1 + a_5^2a_3^2 + 2a_2a_5^2a_4 - a_5^3a + 3)x - a_5^5\end{aligned}$$

6. 矩阵多项式的求解

矩阵多项式的数学形式为

$$\boldsymbol{B} = a_1\boldsymbol{A}^n + a_2\boldsymbol{A}^{n-1} + \cdots + a_n\boldsymbol{A} + a_{n+1}\boldsymbol{I} \tag{3-2-9}$$

其中，$\boldsymbol{A}$ 为一个给定矩阵，$\boldsymbol{I}$ 为和 $\boldsymbol{A}$ 同阶次的单位矩阵，这时返回的矩阵 $\boldsymbol{B}$ 为矩阵多项式的值。矩阵多项式的值在 MATLAB 语言环境中可以由 polyvalm() 函数求出，该函数的调用格式为 B=polyvalm(a,A)，其中，a 为多项式系数降幂排列构成的向量，即 a=[$a_1, a_2, \cdots, a_n, a_{n+1}$]。

相应地，还可以按点运算的方式定义一种多项式运算为

$$\boldsymbol{C} = a_1\boldsymbol{x}.\hat{}n + a_2\boldsymbol{x}.\hat{}(n-1) + \cdots + a_{n+1} \tag{3-2-10}$$

这时，矩阵 $\boldsymbol{C}$ 可以由语句 C=polyval(a,x) 直接计算出来。

若由 MATLAB 的符号运算工具箱给出多项式 p，则可以调用 subs() 函数求出点运算意义下的多项式的值 C=subs(p,s,x)。

例 3-15 Cayley-Hamilton 定理是矩阵理论中的一个比较重要的定理，其内容为：若矩阵 $\boldsymbol{A}$ 的特征多项式为

$$f(s) = \det(s\boldsymbol{I} - \boldsymbol{A}) = a_1s^n + a_2s^{n-1} + \cdots + a_ns + a_{n+1} \tag{3-2-11}$$

则有 $f(\boldsymbol{A}) = \boldsymbol{0}$，亦即 $a_1\boldsymbol{A}^n + a_2\boldsymbol{A}^{n-1} + \cdots + a_n\boldsymbol{A} + a_{n+1}\boldsymbol{I} = \boldsymbol{0}$。假设矩阵 $\boldsymbol{A}$ 为 Vandermonde 矩阵，试验证其满足 Cayley-Hamilton 定理。

求解 可以由下面的 MATLAB 语句来验证 Cayley-Hamilton 定理。

```
>> A=vander([1 2 3 4 5 6 7])
   aa=poly(A); B=polyvalm(aa,A); norm(B)
```

则可以构造出下面的 Vandermonde 矩阵，并求出误差矩阵的范数为 2.1886×10^6。

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 729 & 243 & 81 & 27 & 9 & 3 & 1 \\ 4096 & 1024 & 256 & 64 & 16 & 4 & 1 \\ 15625 & 3125 & 625 & 125 & 25 & 5 & 1 \\ 46656 & 7776 & 1296 & 216 & 36 & 6 & 1 \\ 117649 & 16807 & 2401 & 343 & 49 & 7 & 1 \end{bmatrix}$$

由于使用的 poly() 函数会产生一定的误差，而该误差在矩阵多项式求解中导致了巨大的误差，从而得出错误结论。由此看来，poly() 函数的误差有时是不可忽略的。如果把上面语句中的 poly() 函数用 poly1() 函数代替，则可以得出如下结果：

```
>> aa1=poly1(A); B1=polyvalm(aa1,A); norm(B1)
```

可见，由此得出的 $\boldsymbol{B}$ 矩阵就会等于 0，故该矩阵满足 Cayley-Hamilton 定理。

3.2.2 符号多项式与数值多项式的转换

若已知数值多项式系数构成的向量 p=[$a_1, a_2, \cdots, a_{n+1}$]，则可以通过符号运算工具箱提供的 poly2sym() 函数转换成多项式表示。若已知多项式的符号表

达式，则可以由 sym2poly() 函数转换成系数向量形式。这两个函数的调用格式都是很简单的：

```
f=poly2sym(p) 或 f=poly2sym(p,x)   % 向量转多项式
p=sym2poly(f)                       % 多项式转向量
```

例 3-16 已知多项式 $f=s^5+2s^4+3s^3+4s^2+5s+6$，试用不同形式表示该多项式。

求解 该多项式可以用两种形式先定义出来。例如，可以用数值形式先定义再用相应的方式将其转换成符号型的多项式。

```
>> P=[1 2 3 4 5 6]; % 先由系数按降幂顺序排列表示多项式
   f=poly2sym(P,'v')    % 以 v 为算子表示多项式
```

这样得出的多项式为 $f(v)=v^5+2v^4+3v^3+4v^2+5v+6$。显然，另一种方法是需要先表示符号形式的多项式，然后用转换函数将其转换成数值形式。

```
>> P=sym2poly(f)    % 转换成数值形式的多项式
```

这样会得出向量 $\boldsymbol{p}=[1,2,3,4,5,6]$。

3.2.3 逆矩阵与广义逆矩阵

1. 矩阵的逆矩阵

对一个已知的 $n\times n$ 非奇异方阵 $\boldsymbol{A}$ 来说，若有一个同样大小的 $\boldsymbol{C}$ 矩阵满足

$$\boldsymbol{AC}=\boldsymbol{CA}=\boldsymbol{I} \tag{3-2-12}$$

式中 $\boldsymbol{I}$ 为单位阵，则称 $\boldsymbol{C}$ 矩阵为 $\boldsymbol{A}$ 矩阵的逆矩阵，并记作 $\boldsymbol{C}=\boldsymbol{A}^{-1}$。

MATLAB 语言中提供了 inv() 函数，可以直接用来求取矩阵的逆矩阵，亦即 $\boldsymbol{C}$=inv($\boldsymbol{A}$)。该函数同样适用于符号变量构成的矩阵的求逆。

例 3-17 试求取 Hilbert 矩阵的逆矩阵。

求解 考虑 4×4 Hilbert 矩阵，调用 MATLAB 的矩阵求逆函数 inv()，则可以立即得出该矩阵的逆矩阵。

```
>> H=hilb(4); H1=inv(H), H*H1
```

这样得出的逆矩阵 $\boldsymbol{H}_1$ 为

$$\begin{bmatrix} 15.9999999999993 & -119.999999999992 & 239.999999999979 & -139.999999999986\\ -119.999999999992 & 1199.9999999999 & -2699.99999999976 & 1679.99999999984\\ 239.99999999998 & -2699.99999999976 & 6479.9999999994 & -4199.99999999961\\ -139.999999999987 & 1679.99999999984 & -4199.99999999961 & 2799.99999999974 \end{bmatrix}$$

这里还可以应用 MATLAB 的语句来检验得出的逆矩阵是否符合条件。例如，计算原

矩阵 $\boldsymbol{H}$ 和求出的逆矩阵 $\boldsymbol{H}_1$ 之积，则

$$\boldsymbol{H}\boldsymbol{H}_1=\begin{bmatrix}1 & 2.2737\times10^{-13} & -4.5475\times10^{-13} & 2.2737\times10^{-13}\\ 7.1054\times10^{-15} & 1 & -1.1369\times10^{-13} & 1.1369\times10^{-13}\\ 7.1054\times10^{-15} & 0 & 1 & 0\\ 3.5527\times10^{-15} & 1.1369\times10^{-13} & -1.1369\times10^{-13} & 1\end{bmatrix}$$

二者之积应该为单位阵，但实际上这样的积有点误差。对大规模矩阵来说，如果用上述的检验方法显得太麻烦，所以可以将结果减去一个单位阵，对其误差用求范数的方法，对得出的矩阵进行检验。如果误差矩阵的范数是一个微小的数，则可以接受得出的逆矩阵，否则应该认为其不正确。从结果看，此误差虽然未小于 MATLAB 矩阵运算的一般误差 ($10^{-15}\sim10^{-16}$ 数量级)，但还是比较小的，因此可以接受得出的逆矩阵。

考虑到高阶 Hilbert 矩阵接近于奇异矩阵，一般不建议用 inv() 函数直接求解，可以采用 invhilb() 函数直接产生逆矩阵。对于低阶矩阵，用 invhilb() 计算出来的逆矩阵的精度也显著改善了。若扩大矩阵的阶次，例如需要研究 13×13 的 Hilbert 矩阵，则两种求逆的方法都将失效。解决这样问题的最好方法是采用解析解方法。

符号运算工具箱中也对符号矩阵定义了 inv() 函数，即使对更高阶的非奇异矩阵也可以精确求解出矩阵的逆矩阵来。现在先显示 6×6 的 Hilbert 逆矩阵为

```
>> H=sym(hilb(6)); H1=inv(H)
```

这样求出的精确逆矩阵为

$$\boldsymbol{H}_1=\begin{bmatrix}36 & -630 & 3360 & -7560 & 7560 & -2772\\ -630 & 14700 & -88200 & 211680 & -220500 & 83160\\ 3360 & -88200 & 564480 & -1411200 & 1512000 & -582120\\ -7560 & 211680 & -1411200 & 3628800 & -3969000 & 1552320\\ 7560 & -220500 & 1512000 & -3969000 & 4410000 & -1746360\\ -2772 & 83160 & -582120 & 1552320 & -1746360 & 698544\end{bmatrix}$$

其实，用符号运算工具箱可以求解出更高阶 Hilbert 矩阵的逆矩阵。例如，求解 30 阶矩阵，可以使用下面的命令，得出精确的逆矩阵，这时误差为 0。

```
>> H=sym(hilb(30)); norm(double(H*inv(H)-eye(size(H))))
```

例 3-18 试对例 3-9 中给出的奇异矩阵 $\boldsymbol{A}$ 求逆，并观察用数值方法对真正奇异的矩阵求逆会发生什么现象。

求解 首先输入该矩阵，则可以用 inv() 函数对其求逆。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   B = inv(A), A*B
```

该函数调用将给出如下的警告信息，提示用户该矩阵接近于奇异。

```
Warning: Matrix is close to singular or badly scaled.
```

```
Results may be inaccurate. RCOND = 1.306145e-017.
```

事实上，$\boldsymbol{A}$ 矩阵是奇异矩阵，但通过数值算法后得出的是接近奇异的结论。该提示说明得出的逆矩阵可能是不正确的，得出的“逆”矩阵 $\boldsymbol{B}$ 和 $\boldsymbol{AB}$ 矩阵分别为

$$\boldsymbol{B}=10^{14}\begin{bmatrix}0.9383 & 2.8147 & -2.8147 & -0.9383\\ 2.8147 & 8.4442 & -8.4442 & -2.8147\\ -2.8147 & -8.4442 & 8.4442 & 2.8147\\ -0.9383 & -2.8147 & 2.8147 & 0.9383\end{bmatrix},\ \boldsymbol{AB}=\begin{bmatrix}1 & 0 & -1 & -0.25\\ -0.25 & 0 & 0 & 0.875\\ 0.25 & 0.5 & 0 & 0.25\\ 0.1563 & 0.125 & 0 & 1.734\end{bmatrix}$$

事实上，奇异矩阵根本不存在一个相应的逆矩阵，能满足式 (3-2-12) 中的条件。对这里给出的问题还可以试用符号运算工具箱中的函数，但由于矩阵奇异，故 inv() 函数也无能为力。

```
>> A=sym(A); inv(A)
```

这时将得出错误信息为

```
??? Error using ==> sym/inv
Error, (in inverse) singular matrix
```

例 3-19 MATLAB 的矩阵求逆函数同样适用于含有变量的矩阵。例如，对于下面的 Hankel 矩阵，可以直接用 inv() 函数得出其逆矩阵。

```
>> syms a1 a2 a3 a4; H=hankel([a1 a2 a3 a4]); inv(H)
```

这样得出其逆矩阵为

$$\boldsymbol{H}^{-1}=\begin{bmatrix}0 & 0 & 0 & 1/a_4\\ 0 & 0 & 1/a_4 & -1/a_4^2a_3\\ 0 & 1/a_4 & -1/a_4^2a_3 & -1/a_4^3(a_2a_4-a_3^2)\\ 1/a_4 & -1/a_4^2a_3 & -1/a_4^3(a_2a_4-a_3^2) & (-a_1a_4^2+2a_2a_3a_4-a_3^3)/a_4^4\end{bmatrix}$$

2. 矩阵的广义逆

前面已经介绍过，即使用解析解求解的符号运算工具箱对奇异矩阵的求逆也是无能为力的，因为其逆矩阵根本不存在。另外，长方形的矩阵有时也会涉及到不能求逆的问题，这样就需要定义一种新的“逆矩阵”。对于要研究的矩阵 $\boldsymbol{A}$，如果存在一个矩阵 $\boldsymbol{N}$，满足

$$\boldsymbol{ANA}=\boldsymbol{A} \tag{3-2-13}$$

则 $\boldsymbol{N}$ 矩阵称为 $\boldsymbol{A}$ 的广义逆矩阵，记作 $\boldsymbol{N}=\boldsymbol{A}^{-}$。如果 $\boldsymbol{A}$ 矩阵是一个 $n\times m$ 的长方形矩阵，则 $\boldsymbol{N}$ 矩阵为 $m\times n$ 阶矩阵。满足该条件的广义逆矩阵有无穷多个。

定义下面的范数最小化指标为

$$\min_{\boldsymbol{x}}||\boldsymbol{Ax}-\boldsymbol{B}|| \tag{3-2-14}$$

则可以证明，对于一个给定的矩阵 $\boldsymbol{A}$，存在一个惟一的矩阵 $\boldsymbol{M}$ 使得下面的 3 个条件同时成立。

① $\boldsymbol{AMA}=\boldsymbol{A}$

② $\boldsymbol{MAM}=\boldsymbol{M}$

③ $\boldsymbol{AM}$ 与 $\boldsymbol{MA}$ 均为 Hermite 对称矩阵

这样的矩阵 $\boldsymbol{M}$ 称为矩阵 $\boldsymbol{A}$ 的 Moore-Penrose 广义逆矩阵，记作 $\boldsymbol{M}=\boldsymbol{A}^+$。从上面的 3 个条件中可以看出，第一个条件和一般广义逆的定义也是一样的，所不同的是它还要求满足第二个和第三个条件，这样将得出惟一的广义逆矩阵了。

MATLAB 提供了求取矩阵 Moore-Penrose 广义逆的函数 pinv()。该函数的调用格式为

```
M=pinv(A)      % 按默认精度求取 Moore-Penrose 广义逆
M=pinv(A,ϵ)    % 按指定精度 ϵ 求解广义逆矩阵
```

其中，ϵ 为判 0 用误差限，如果省略此参数，则判 0 用误差限选用机器的精度 eps，这时将返回 $\boldsymbol{A}$ 的 Moore-Penrose 广义逆矩阵 $\boldsymbol{M}$。如果 $\boldsymbol{A}$ 矩阵为非奇异方阵，则该函数得出的结果就是矩阵的逆阵，但这样求解的速度将明显慢于 inv() 函数。

例 3-20 考虑例 3-9 中给出的奇异矩阵 $\boldsymbol{A}$，例 3-18 中用符号运算工具箱中 inv() 函数仍不能获得问题的解析解，因为解析解不存在。所以这里将考虑 Moore-Penrose 广义逆矩阵的求解。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   B=pinv(A), A*B
```

这样可以得出广义逆矩阵 $\boldsymbol{B}$，并求出 $\boldsymbol{AB}$ 如下

$$\boldsymbol{B}=\begin{bmatrix}0.1011 & -0.0739 & -0.0614 & 0.0636\\ -0.0364 & 0.0386 & 0.0261 & 0.0011\\ 0.0136 & -0.0114 & -0.0239 & 0.0511\\ -0.0489 & 0.0761 & 0.0886 & -0.0864\end{bmatrix},\ \boldsymbol{AB}=\begin{bmatrix}0.95 & -0.15 & 0.15 & 0.05\\ -0.15 & 0.55 & 0.45 & 0.15\\ 0.15 & 0.45 & 0.55 & -0.15\\ 0.05 & 0.15 & -0.15 & 0.95\end{bmatrix}$$

这个结果就不再是单位阵了，因为不存在一个 $\boldsymbol{A}^+$ 能使它成为单位阵。这样得出的 $\boldsymbol{A}^+$ 应该能使得式 (3-2-14) 中的范数取最小值。现在检验 Moore-Penrose 广义逆的 3 个条件如下：

```
>> norm(A*B*A-A), norm(B*A*B-B), norm(A*B-B'*A'), norm(B*A-A'*B')
```

这 4 个误差矩阵范数分别为 $3.3781\times10^{-14}, 9.3458\times10^{-17}, 1.3651\times10^{-15}, 1.6555\times10^{-15}$。由此证实得出的逆矩阵确实是原矩阵的 Moore-Penrose 广义逆。现在对得出的 $\boldsymbol{B}$ 再求一次 Moore-Penrose 广义逆，则可看出

$$(\boldsymbol{A}^+)^+=\begin{bmatrix}16 & 1.99999999999999 & 3 & 13\\ 5 & 11 & 10 & 8.00000000000001\\ 8.99999999999998 & 6.99999999999999 & 6 & 12\\ 4 & 14 & 15 & 1.00000000000001\end{bmatrix}=\boldsymbol{A}$$

例 3-21 考虑给定的长方形矩阵 $A=\begin{bmatrix}6 & 1 & 4 & 2 & 1\\ 3 & 0 & 1 & 4 & 2\\ -3 & -2 & -5 & 8 & 4\end{bmatrix}$，试对该矩阵进行基本分析，例如获得矩阵的秩、Moore-Penrose 广义逆等，并分析得出的广义逆矩阵性质。

求解 可以给出下面的语句对该矩阵进行分析，则可见该矩阵的秩为 2，故原矩阵为非满秩矩阵。

```
>> A=[6,1,4,2,1; 3,0,1,4,2; -3,-2,-5,8,4]; rank(A)
```

由于 A 矩阵为奇异矩阵，所以应该使用 pinv() 函数求取矩阵的 Moore-Penrose 广义逆，并可用通过下面的检验语句对 Moore-Penrose 广义逆的条件逐一验证，证实该广义逆矩阵确实满足条件。

```
>> iA = pinv(A), norm(iA*A*iA-iA), % 非满秩矩阵的广义逆
   norm(A*iA*A-A), norm(iA*A-A'*iA'), norm(A*iA-iA'*A')
```

广义逆矩阵为 $A^+=\begin{bmatrix}0.073025 & 0.041301 & -0.022147\\ 0.010774 & 0.0019952 & -0.015563\\ 0.04589 & 0.017757 & -0.038508\\ 0.032721 & 0.043097 & 0.063847\\ 0.016361 & 0.021548 & 0.031923\end{bmatrix}$，并得出 4 个误差矩阵的范数分别为

$$||A^+AA^+-A^+||=1.0264\times10^{-16},\ ||AA^+A-A||=8.1145\times10^{-15}$$
$$||A^+A-A^{\rm T}(A^+)^{\rm T}||=3.9098\times10^{-16},\ ||AA^+-(A^+)^{\rm T}A^{\rm T}||=1.6653\times10^{-16}$$

3.2.4 矩阵的特征值问题

1. 一般矩阵的特征值与特征向量

对一个矩阵 A 来说，如果存在一个非零的向量 x，且有一个标量 λ 满足

$$Ax=\lambda x \tag{3-2-15}$$

则称 λ 为 A 矩阵的一个特征值，而 x 称为对应于特征值 λ 的特征向量。严格说来，x 应该称为 A 的右特征向量。如果矩阵 A 的特征值不包含重复的值，则对应的各个特征向量为线性无关的，这样由各个特征向量可以构成一个非奇异的矩阵。如果用它对原始矩阵作相似变换，则可以得出一个对角矩阵。矩阵的特征值与特征向量由 MATLAB 提供的函数 eig() 可以容易地求出。该函数的调用格式为

```
d=eig(A)          % 只求解特征值
[V, D]=eig(A)     % 求解特征值和特征向量
```

其中，d 特征值构成的向量，D 为一个对角矩阵，其对角线上的元素为矩阵 A 的特征值，而每个特征值对应的 V 矩阵的列为该特征值的特征向量，该矩阵是一个满秩矩阵。MATLAB 的矩阵特征值的结果满足 $AV=VD$，且每个特征向量各

元素的平方和 (即 2 范数) 均为 1。如果调用该函数时只给出一个返回变量，则将只返回矩阵 $\boldsymbol{A}$ 的特征值。即使 $\boldsymbol{A}$ 为复数矩阵，也照样可以由 eig() 函数得出其特征值与特征向量矩阵的。

前面介绍的矩阵特征多项式的根和特征值是同样的概念，所以若精确已知矩阵的特征多项式系数，则可以调用 roots() 函数来计算矩阵的特征值。

矩阵特征值的求解算法是多种多样的，最常用的有求解实对称矩阵特征值与特征向量的 Jacobi 算法、原点平移 QR 分解法与两步 QR 算法[6]。矩阵的特征值与特征向量的求解有许多标准的子程序或程序库可以直接调用，如著名的 EISPACK 软件包[7,8] 等。MATLAB 中的 eig() 函数是基于两步 QR 算法实现的，该函数也同样可以求解复数矩阵的特征值与特征向量矩阵。当矩阵含有重特征值时，特征向量矩阵可能趋于奇异，所以在使用此函数时应该注意。

例 3-22 求出例 3-9 中给出的矩阵 $\boldsymbol{A}$ 的特征值与特征向量矩阵。

求解 可以调用 eig() 函数直接获得矩阵 $\boldsymbol{A}$ 的特征值。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; eig(A)
```

得出的特征值为 34, 8.9443, −8.9443, 0。符号运算工具箱中也提供了 eig() 函数，理论上可以求解任意高阶矩阵的精确特征值，对于给定的 $\boldsymbol{A}$ 矩阵，可以由 eig(sym(A)) 命令求出特征值的精确解为 $0, 34, \pm 4\sqrt{5}$。

如果想求出高精度的数值解，则可以给出如下命令 vpa(ans,70)。

对于数值矩阵 $\boldsymbol{A}$，可以通过下面的语句同时求出矩阵的特征值和特征向量为

```
>> [v,d]=eig(A)
```

得出的特征向量矩阵和特征值矩阵分别为

$$\boldsymbol{v}=\begin{bmatrix}-0.5 & -0.8236 & 0.3764 & -0.2236\\ -0.5 & 0.4236 & 0.0236 & -0.6708\\ -0.5 & 0.0236 & 0.4236 & 0.6708\\ -0.5 & 0.3764 & -0.8236 & 0.2236\end{bmatrix},\ \boldsymbol{d}=\begin{bmatrix}34 & 0 & 0 & 0\\ 0 & 8.9443 & 0 & 0\\ 0 & 0 & -8.9443 & 0\\ 0 & 0 & 0 & 6.67\times 10^{-17}\end{bmatrix}$$

同样，用符号运算工具箱中的 $[\boldsymbol{v},\boldsymbol{d}]$=eig(sym($\boldsymbol{A}$)) 函数也可以求解出特征值和特征向量矩阵的解析解为

$$\boldsymbol{v}=\begin{bmatrix}-8\sqrt{5}-17 & 8\sqrt{5}-17 & 1 & -1\\ 4\sqrt{5}+9 & -4\sqrt{5}+9 & 1 & -3\\ 1 & 1 & 1 & 3\\ 4\sqrt{5}+7 & -4\sqrt{5}+7 & 1 & 1\end{bmatrix},\ \boldsymbol{d}=\begin{bmatrix}4\sqrt{5} & 0 & 0 & 0\\ 0 & -4\sqrt{5} & 0 & 0\\ 0 & 0 & 34 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

可见，在前面的例子中两次调用了 eig() 函数，但由于返回参数个数不一致，所以前面的调用返回矩阵 $\boldsymbol{A}$ 的特征值与特征向量，而后面的调用只返回矩阵 $\boldsymbol{A}$ 的特征值而不返回特征向量矩阵。另外，返回特征值的格式也因返回变量个数不同而不同。

如果一个矩阵包含重特征值，则理论上矩阵 $\boldsymbol{V}$ 将为奇异矩阵。但因为 MATLAB 数值运算出现的误差，不一定能精确计算出矩阵的重根，这样将得出接近奇异的 $\boldsymbol{V}$ 矩阵。

若想由 C 或 Fortran 语句从最底层编程，则需要编写相当大的程序才可以完成特征值和特征向量的计算，例如，EISPACK [8] 中提供的子程序源程序有 500 多条语句。该子程序提供的算法是数值的，不能进行解析运算。

2. 矩阵的广义特征向量问题

若某矩阵 $\boldsymbol{A}$ 含有重特征值，则必定会使得特征向量矩阵为奇异矩阵，这会约束特征向量矩阵的应用。为了保证特征向量矩阵非奇异，需要引入广义特征向量的问题。假设存在一个标量 λ 和一个非零向量 $\boldsymbol{x}$，使得

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x} \tag{3-2-16}$$

成立，其中 $\boldsymbol{B}$ 矩阵为对称正定矩阵，则 λ 称为广义特征值，而 $\boldsymbol{x}$ 向量称为广义特征向量。MATLAB 还提供了求取广义特征值的方法。事实上，普通的矩阵特征值问题可以看成是广义特征值问题的一个特例，因为若假定 $\boldsymbol{B} = \boldsymbol{I}$ 为单位阵，则式 (3-2-16) 中的形式可以直接转化成普通矩阵特征值问题。

文献 [9] 中给出了广义特征值问题的 QZ 算法。在 MATLAB 中给出的 `eig()` 函数可以直接用来求取矩阵的广义特征值和特征向量，这时的调用格式为

```
d=eig(A,B)          % 求解广义特征值
[V,D]=eig(A,B)      % 求解广义特征值和特征向量
```

这一函数可以直接得出矩阵的广义特征值向量 $\boldsymbol{d}$，也可以返回一个特征向量矩阵 $\boldsymbol{V}$ 及一个对角型特征值矩阵 $\boldsymbol{D}$，满足 $\boldsymbol{AV} = \boldsymbol{BVD}$。值得指出的是，该函数可以求解 $\boldsymbol{B}$ 矩阵为奇异矩阵时的广义特征值问题。

例 3-23 假设给出如下的矩阵

$$\boldsymbol{A} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 2 & 6 & -1 & -2 \\ 5 & -1 & 2 & 3 \\ -3 & -4 & 1 & 10 \\ 5 & -2 & -3 & 8 \end{bmatrix}$$

请求出 $\boldsymbol{A}, \boldsymbol{B}$ 矩阵的广义特征值与特征向量矩阵。

求解 使用下列命令可以求出矩阵的广义特征值和特征向量。

```
>> A=[5,7,6,5; 7,10,8,7; 6,8,10,9; 5,7,9,10];
   B=[2,6,-1,-2; 5,-1,2,3; -3,-4,1,10; 5,-2,-3,8];
   [V,D]=eig(A,B)
```

这样可以得出特征值和特征向量矩阵分别为

$$\boldsymbol{D}=\begin{bmatrix}4.7564 & 0 & 0 & 0\\ 0 & 0.047055+0.17497\mathrm{j} & 0 & 0\\ 0 & 0 & 0.047055-0.17497\mathrm{j} & 0\\ 0 & 0 & 0 & -0.003689\end{bmatrix}$$

$$\boldsymbol{V}=\begin{bmatrix}0.3697 & -0.37409+0.62591\mathrm{j} & -0.37409-0.62591\mathrm{j} & 1\\ 0.99484 & -0.067434-0.25314\mathrm{j} & -0.067434+0.25314\mathrm{j} & -0.60903\\ 0.79792 & 0.92389+0.026381\mathrm{j} & 0.92389-0.026381\mathrm{j} & -0.23164\\ 1 & -0.65986-0.32628\mathrm{j} & -0.65986+0.32628\mathrm{j} & 0.13186\end{bmatrix}$$

符号运算工具箱中的 `eig()` 函数不支持广义特征值的运算。

3. 复数矩阵的特征值与 Gershgorin 定理

当然，任意复数矩阵的特征值可以由 MATLAB 中的 `eig()` 函数精确求解。这里探讨的内容虽然不能直接用于复数矩阵特征值的精确求解，但由该方法引出了一个新的多变量系统分析与计算机辅助设计的领域 —— 多变量系统的逆 Nyquist 阵列频域分析。

Gershgorin 定理是基于逆 Nyquist 阵列的多变量方法的核心。对复数矩阵

$$\boldsymbol{C}=\begin{bmatrix}c_{11} & \cdots & c_{1k} & \cdots & c_{1n}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ c_{k1} & \cdots & c_{kk} & \cdots & c_{kn}\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ c_{n1} & \cdots & c_{nk} & \cdots & c_{nn}\end{bmatrix} \tag{3-2-17}$$

来说，矩阵的特征值 λ 满足

$$|\lambda-c_{kk}|\leqslant\sum_{j\neq k}|c_{kj}|,\ \text{且}\ |\lambda-c_{kk}|\leqslant\sum_{j\neq k}|c_{jk}| \tag{3-2-18}$$

换句话说，该矩阵的特征值位于一族以 c_{kk} 为圆心，以不等式右面的表达式为半径的圆构成的并集内，而这些圆又称为 Gershgorin 圆。另外，上面两个不等式表示的关系分别称为列 Gershgorin 圆和行 Gershgorin 圆。

其实，对传统的 Gershgorin 定理直接拓展，就可能得出更小半径的圆：

$$|\lambda-c_{kk}|\leqslant\min\left(\sum_{j\neq k}|c_{kj}|,\ \sum_{j\neq k}|c_{jk}|\right) \tag{3-2-19}$$

3.2.5　矩阵的 Kronecker 乘积

矩阵的 Kronecker 乘积在一些特殊矩阵方程求解中有其独特的作用。假设 $\boldsymbol{A}$ 矩阵为 $n \times m$ 矩阵，则矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的 Kronecker 乘积如下定义

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{m1}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\boldsymbol{B} & \cdots & a_{nm}\boldsymbol{B} \end{bmatrix} \tag{3-2-20}$$

MATLAB 提供了 `kron()` 函数，可以计算 $\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B}$，该函数的调用格式为 $\boldsymbol{C}$ =`kron(`$\boldsymbol{A}$`,`$\boldsymbol{B}$`)`。任意维数的矩阵均可计算 Kronecker 乘积。

例 3-24　假设 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵如下给出，试求出 $\boldsymbol{A} \otimes \boldsymbol{B}$ 和 $\boldsymbol{B} \otimes \boldsymbol{A}$。

$$\boldsymbol{A} = \begin{bmatrix} 7 & 5 & 5 \\ 4 & 4 & 2 \\ 9 & 8 & 7 \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} 2 & 6 & 2 \\ 2 & 3 & 0 \end{bmatrix}$$

求解　可以输入这两个矩阵，然后得出所需的 Kronecker 乘积矩阵

```
>> A=[7,5,5; 4,4,2; 9,8,7]; B=[2,6,2; 2,3,0];
   C1=kron(A,B), C2=kron(B,A)
```

两个乘积矩阵如下，可见，二者是不相同的。

$$\boldsymbol{C}_1 = \left[\begin{array}{ccc:ccc:ccc} 14 & 42 & 14 & 10 & 30 & 10 & 10 & 30 & 10 \\ 14 & 21 & 0 & 10 & 15 & 0 & 10 & 15 & 0 \\ \hdashline 8 & 24 & 8 & 8 & 24 & 8 & 4 & 12 & 4 \\ 8 & 12 & 0 & 8 & 12 & 0 & 4 & 6 & 0 \\ \hdashline 18 & 54 & 18 & 16 & 48 & 16 & 14 & 42 & 14 \\ 18 & 27 & 0 & 16 & 24 & 0 & 14 & 21 & 0 \end{array}\right],\quad \boldsymbol{C}_2 = \left[\begin{array}{ccc:ccc:ccc} 14 & 10 & 10 & 42 & 30 & 30 & 14 & 10 & 10 \\ 8 & 8 & 4 & 24 & 24 & 12 & 8 & 8 & 4 \\ \hdashline 18 & 16 & 14 & 54 & 48 & 42 & 18 & 16 & 14 \\ 14 & 10 & 10 & 21 & 15 & 15 & 0 & 0 & 0 \\ \hdashline 8 & 8 & 4 & 12 & 12 & 6 & 0 & 0 & 0 \\ 18 & 16 & 14 & 27 & 24 & 21 & 0 & 0 & 0 \end{array}\right]$$

3.2.6　矩阵微积分运算

前面介绍的微积分问题都是标量函数的微积分问题，可以由符号运算工具箱中的 `diff()` 和 `int()` 函数直接求解。对给定函数为矩阵函数的情况，则需要按照下面几种情况单独讨论。

1. 单变量函数的微积分

假设矩阵 $\boldsymbol{A}(t)$ 的每个元素 $a_{i,j}(t)$ 都是 t 的函数，则 $\boldsymbol{A}(t)$ 矩阵的微分定义为由每个元素对 t 微分构成的矩阵。矩阵 $\boldsymbol{A}(t)$ 的积分定义也是一样的。

MATLAB 的 `diff()` 和 `int()` 函数可以直接用于矩阵的微分和积分。

例 3-25　假设已知矩阵 $\boldsymbol{A}(t) = \begin{bmatrix} t^3 & 3t & 4\mathrm{e}^{-5t}\sin t \\ 1 & \cos 5t & \sin at \end{bmatrix}$，试求 $\dot{\boldsymbol{A}}(t)$，将其积分，观察是否能还原 $\boldsymbol{A}(t)$ 矩阵。

求解 申明 a,t 为符号变量，则可以立即定义出 $\boldsymbol{A}(t)$ 矩阵，并利用符号运算工具箱的 diff() 和 int() 函数直接求出矩阵的微分和积分矩阵

```
>> syms a t; A=[t^3,3*t,4*exp(-5*t)*sin(t); 1,cos(5*t),sin(a*t)];
   A1=diff(A), A2=int(A1)
```

得出的结果为

$$\boldsymbol{A}_1(t)=\begin{bmatrix}3t^2 & 3 & -20\mathrm{e}^{-5t}\sin t+4\mathrm{e}^{-5t}\cos t\\ 0 & -5\sin 5t & a\cos at\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}t^3 & 3t & 4\mathrm{e}^{-5t}\sin t\\ 0 & \cos 5t & \sin at\end{bmatrix}$$

可见，矩阵的微分 $\boldsymbol{A}_1(t)$ 可以立即得出，对其求积分则可以得出 $\boldsymbol{A}_2(t)$ 矩阵。可见，除了原 $\boldsymbol{A}(t)$ 矩阵的常数项外，其余各项均能还原。

2. 多元函数对矩阵的微积分

假设已知多变量标量函数 $f(\boldsymbol{X})$，其中 $\boldsymbol{X}$ 为 $n\times m$ 矩阵，则 $\mathrm{d}f(\boldsymbol{X})/\mathrm{d}\boldsymbol{X}$ 的定义为

$$\frac{\mathrm{d}f(\boldsymbol{X})}{\mathrm{d}\boldsymbol{X}}=\begin{bmatrix}\mathrm{d}f/\mathrm{d}x_{11} & \mathrm{d}f/\mathrm{d}x_{12} & \cdots & \mathrm{d}f/\mathrm{d}x_{1m}\\ \mathrm{d}f/\mathrm{d}x_{21} & \mathrm{d}f/\mathrm{d}x_{22} & \cdots & \mathrm{d}f/\mathrm{d}x_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ \mathrm{d}f/\mathrm{d}x_{n1} & \mathrm{d}f/\mathrm{d}x_{n2} & \cdots & \mathrm{d}f/\mathrm{d}x_{nm}\end{bmatrix} \tag{3-2-21}$$

例 3-26 考虑二次型函数 $f(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}-2\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}$，试求解该函数的极值问题[10]。

求解 假设 $\boldsymbol{A}=\{a_{ij}\}$，$\boldsymbol{b}^{\mathrm{T}}=[b_1,\cdots,b_n]$，则二次型函数可以写成

$$\begin{aligned}f(\boldsymbol{x})=&\ a_{11}x_1^2+a_{22}x_2^2+\cdots+a_{nn}x_n^2\\ &+(a_{12}+a_{21})x_1x_2+(a_{13}+a_{31})x_1x_3+\cdots+(a_{n,n-1}+a_{n,n-1})x_nx_{n-1}\\ &-2b_1x_1-2b_2x_2-\cdots-2b_nx_n\end{aligned}$$

由高等数学已知，若一个函数有极值，则该函数对各个自变量的导数都等于零。对给定的函数求导，则

$$\frac{\mathrm{d}f(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}}=\begin{bmatrix}2a_{11}x_1+(a_{12}+a_{21})x_2+\cdots+(a_{1n}+a_{n1})x_n-2b_1\\ (a_{21}+a_{12})x_1+2a_{22}x_2+\cdots+(a_{2n}+a_{n2})x_n-2b_2\\ \vdots\\ (a_{n1}+a_{1n})x_1+(a_{n2}+a_{2n})x_2+\cdots+2a_{nn}x_n-2b_n\end{bmatrix}=\boldsymbol{0}$$

求解该方程则将得出 x_i 的值，这些值相对于函数 $f(\boldsymbol{x})$ 的极值点。特别地，若矩阵 $\boldsymbol{A}$ 为对称矩阵，则 $a_{ij}+a_{ji}=2a_{ij}=2a_{ji}$，故向量 $\boldsymbol{x}$ 可以由 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ 直接求出。

3. 多元函数的 Jacobi 矩阵

假设有 n 个自变量的 m 个函数定义为

$$\begin{cases} y_1 = f_1(x_1, x_2, \cdots, x_n) \\ y_2 = f_2(x_1, x_2, \cdots, x_n) \\ \quad\vdots \\ y_m = f_m(x_1, x_2, \cdots, x_n) \end{cases} \tag{3-2-22}$$

则相应的 y_i 对 x_j 求偏导，则得出矩阵

$$\boldsymbol{J} = \begin{bmatrix} \partial y_1/\partial x_1 & \partial y_1/\partial x_2 & \cdots & \partial y_1/\partial x_n \\ \partial y_2/\partial x_1 & \partial y_2/\partial x_2 & \cdots & \partial y_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial y_m/\partial x_1 & \partial y_m/\partial x_2 & \cdots & \partial y_m/\partial x_n \end{bmatrix} \tag{3-2-23}$$

该矩阵又称为 Jacobi 矩阵，它在图像处理、机器人等诸多领域中均是很有用的概念。Jacobi 矩阵可以由 MATLAB 的符号运算工具箱中的 `jacobian()` 函数直接求得。该函数的调用格式为 `J=jacobian(y,x)`，其中，$\boldsymbol{x}$ 是自变量构成的向量，$\boldsymbol{y}$ 是由各个函数构成的向量。

例 3-27 假设有直角坐标和极坐标变换公式为 $x = r\sin\theta\cos\phi$, $y = r\sin\theta\sin\phi$, $z = r\cos\theta$，试推导其 Jacobi 矩阵。

求解 可以先申明 3 个符号变量并描述 3 个函数，这样可以用下面的语句容易地求解出其 Jacobi 矩阵。

```
>> syms r theta phi; x=r*sin(theta)*cos(phi);
   y=r*sin(theta)*sin(phi); z=r*cos(theta);
   J=jacobian([x; y; z],[r theta phi])
```

可以得出 $\boldsymbol{J} = \begin{bmatrix} \sin\theta\cos\phi & r\cos\theta\cos\phi & -r\sin\theta\sin\phi \\ \sin\theta\sin\phi & r\cos\theta\sin\phi & r\sin\theta\cos\phi \\ \cos\theta & -r\sin\theta & 0 \end{bmatrix}$。

3.2.7 矩阵分析在控制理论研究中的应用举例

本节将针对控制理论中若干个重要问题介绍矩阵分析的基本应用。例如，控制系统的可控性和可观测性的判定是基于判定矩阵求秩结果的，状态方程到传递函数转换是矩阵求逆的，线性系统稳定性分析是基于矩阵或多项式特征值求解问题的，系统的极点配置是基于矩阵的综合运算的，所以矩阵分析问题在控制理论研究中是很重要的。本节将通过一些例子介绍这些问题的求解方法。

1. 状态空间的可控性、可观测性

线性系统的可控性和可观测性是基于状态方程的控制理论的基础，可控性和可观测性的概念是 Kalman 于 1960 年提出的[11]，这些性质为系统的状态反馈设计、观测器的设计等提供了依据。假设系统由状态方程 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$ 给出，对任意的初始时刻 t_0，如果状态空间中任一状态 $x_i(t)$ 可以从初始状态 $x_i(t_0)$ 处，由有界的输入信号 $\boldsymbol{u}(t)$ 的驱动下，在有限时间 $t_{\rm f}$ 内能够到达任意预先指定的状态 $x_i(t_{\rm f})$，则称此状态是可控的。如果系统中所有的状态都是可控的，则称该系统为完全可控的系统。

假设系统由状态方程 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$ 给出，对任意的初始时刻 t_0，如果状态空间中任一状态 $x_i(t)$ 在任意有限时刻 $t_{\rm f}$ 的状态 $x_i(t_{\rm f})$ 可以由输出信号在这一时间区间内 $t\in[t_0,t_{\rm f}]$ 的值精确地确定出来，则称此状态是可观测的。如果系统中所有的状态都是可观测的，则称该系统为完全可观测的系统。

系统的可控性和可观测性的最简单测试方法是可控性矩阵和可观测性矩阵的求秩，可控性矩阵和可观测性矩阵可以分别如下构造

$$\boldsymbol{T}_{\rm c}=\left[\boldsymbol{B},\boldsymbol{A}\boldsymbol{B},\boldsymbol{A}^2\boldsymbol{B},\cdots,\boldsymbol{A}^{n-1}\boldsymbol{B}\right],\ \boldsymbol{T}_{\rm o}=\begin{bmatrix}\boldsymbol{C}\\ \boldsymbol{C}\boldsymbol{A}\\ \boldsymbol{C}\boldsymbol{A}^2\\ \vdots\\ \boldsymbol{C}\boldsymbol{A}^{n-1}\end{bmatrix} \tag{3-2-24}$$

则系统的可控性和可观测性可以分别由 `rank(`$\boldsymbol{T}_{\rm c}$`)` 和 `rank(`$\boldsymbol{T}_{\rm o}$`)` 直接判定。

例 3-28 判定下面离散状态方程模型的可控性。

$$\boldsymbol{x}[(k+1)T]=\begin{bmatrix}-2.2 & -0.7 & 1.5 & -1\\ 0.2 & -6.3 & 6 & -1.5\\ 0.6 & -0.9 & -2 & -0.5\\ 1.4 & -0.1 & -1 & -3.5\end{bmatrix}\boldsymbol{x}(kT)+\begin{bmatrix}6 & 9\\ 4 & 6\\ 4 & 4\\ 8 & 4\end{bmatrix}\boldsymbol{u}(kT)$$

求解 可以通过下面的 MATLAB 语句将系统的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵输入到 MATLAB 的工作空间，这样就可以用下面的语句直接判定系统的可控性。

```
>> A=[-2.2,-0.7,1.5,-1; 0.2,-6.3,6,-1.5; ...
      0.6,-0.9,-2,-0.5; 1.4,-0.1,-1,-3.5];
   B=[6,9; 4,6; 4,4; 8,4]; Tc=ctrb(A,B)
```

生成可控性判定矩阵

$$\boldsymbol{T}_{\rm c}=\begin{bmatrix}6 & 9 & -18 & -22 & 54 & 52 & -162 & -118\\ 4 & 6 & -12 & -18 & 36 & 58 & -108 & -202\\ 4 & 4 & -12 & -10 & 36 & 26 & -108 & -74\\ 8 & 4 & -24 & -6 & 72 & 2 & -216 & 34\end{bmatrix}$$

```
>> Tc1=[B,A*B, A^2*B,A^3*B]; % 或用直接方法建立可控性判定矩阵
   rank(Tc) % 判定系统的可控性，因为可得秩为 3，所以系统不可控
```

2. 状态方程到传递函数的转换

对状态方程式 (3-1-8) 两端同时作 Laplace 变换，则可以得出

$$\begin{cases} s\boldsymbol{I}\boldsymbol{X}(s)=\boldsymbol{A}\boldsymbol{X}(s)+\boldsymbol{B}\boldsymbol{U}(s) \\ \boldsymbol{Y}(s)=\boldsymbol{C}\boldsymbol{X}(s)+\boldsymbol{D}\boldsymbol{U}(s) \end{cases} \tag{3-2-25}$$

式中 $\boldsymbol{I}$ 为单位矩阵，其阶次与矩阵 $\boldsymbol{A}$ 相同。这样从式 (3-1-8) 可以得出

$$\boldsymbol{X}(s)=(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{U}(s) \tag{3-2-26}$$

可以由下面的式子得出等效的系统传递函数矩阵模型为

$$\boldsymbol{G}(s)=\boldsymbol{Y}(s)\boldsymbol{U}^{-1}(s)=\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D} \tag{3-2-27}$$

MATLAB 的控制系统工具箱提供了 G_1=tf(G) 函数，可以直接用来求取系统的传递函数模型。

例 3-29 试求出例 3-8 中给出的多变量状态方程模型对应的传递函数矩阵。

求解 由下面语句可以立即得出传递函数矩阵

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
       6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5]; C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D);    % 状态方程模型
   G1=tf(G) % 显示结果从略
```

由得出的结果可以按数学形式将传递函数矩阵改写成

$$\boldsymbol{G}(s)=\begin{bmatrix} \dfrac{3.5s^3-144.1s^2-20.69s-0.8372}{s^4+s^3+0.35s^2+0.05s+0.0024} & \dfrac{0.95s^3-64.13s^2-9.161s-0.374}{s^4+s^3+0.35s^2+0.05s+0.0024} \\ \dfrac{1.15s^3-36.32s^2-6.225s-0.1339}{s^4+s^3+0.35s^2+0.05s+0.0024} & \dfrac{0.85s^3-15.71s^2-2.619s-0.04559}{s^4+s^3+0.35s^2+0.05s+0.0024} \end{bmatrix}$$

3. 状态空间系统的稳定性

连续状态空间的稳定性要求 $\boldsymbol{A}$ 矩阵没有 s 右半平面的特征值。离散系统则要求 $\boldsymbol{F}$ 矩阵不含有单位圆外的特征值。

由于历史局限性，经典控制中分析系统稳定性的方法主要采用间接方法，如连续系统的 Routh-Hurwitz 判据，离散系统的 Jury 判据[12]等，这些判据需要构造响应的表格，比较麻烦，且不直观。随着计算机技术的飞速发展，特别是随着数

学语言如 MATLAB 的日益普及，求解一个系统的特征根轻而易举，所以无需再使用间接方法判定系统的稳定性，由 **eig**(G) 或 **pole**(G) 即可以求出系统全部特征根，根据特征根的实际分布就能直接判定系统稳定性。对离散系统来说，用 abs() 函数可以得出全部特征根的幅值，若存在幅值大于 1 的特征根，则离散系统不稳定。

MATLAB 函数的 **pzmap**(G) 可以绘制出系统全部零极点的位置，离散系统还可以同时绘制出单位圆，并由极点和单位圆的关系判定系统的稳定性。

例 3-30 假设有开环高阶系统的传递函数

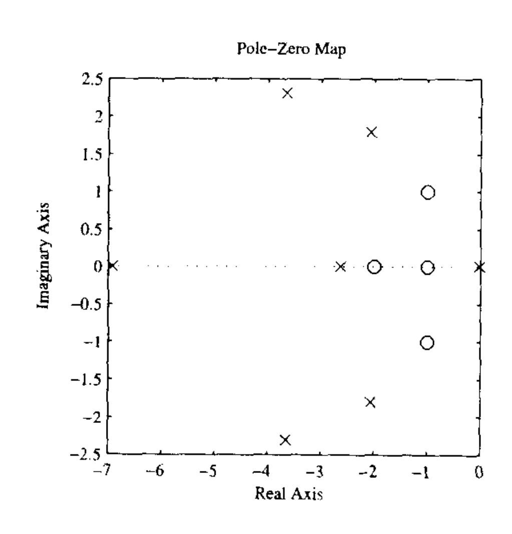
$$G(s)=\frac{10s^4+50s^3+100s^2+100s+40}{s^7+21s^6+184s^5+870s^4+2384s^3+3664s^2+2496s}$$

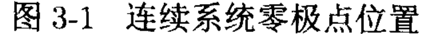
试分析单位负反馈下闭环系统的稳定性。

求解 通过下面的 MATLAB 语句输入系统的传递函数模型并得出单位负反馈构成的闭环系统模型，还可以立即求出系统的全部闭环极点

```
>> num=[10,50,100,100,40]; den=[1,21,184,870,2384,3664,2496,0];
   G=tf(num,den); GG=feedback(G,1); % 输入开环模型并得出闭环模型
   pzmap(GG)  % 用图形显示系统的全部的零极点位置
   eig(GG) % 显示系统的极点，为节省版面，这里只给出其值
```

闭环系统的极点为 $-6.9223, -3.6502\pm \mathrm{j}2.302, -2.0633\pm \mathrm{j}1.7923, -2.6349, -0.015765$，因为该系统全部极点都在 s 左半平面，故此闭环系统是稳定的。图 3-1 中显示的极点位置也证实了上面的结论。




图 3-1 连续系统零极点位置

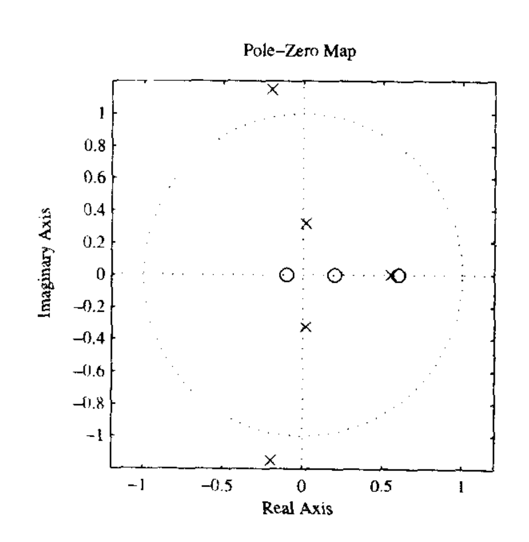


图 3-2 离散闭环系统零极点位置

其实，采用零极点变换语句 zpk(GG) 可以得出如下的零极点模型

$$G(s)=\frac{10(s+2)(s+1)(s^2+2s+2)}{(s+6.922)(s+2.635)(s+0.01577)(s^2+4.127s+7.47)(s^2+7.3s+18.62)}$$

例 3-31 假设离散系统的受控对象传递函数为 $H(z)=\dfrac{6z^2-0.6z-0.12}{z^4-z^3+0.25z^2+0.25z-0.125}$，采样周期为 $T=0.1$ 秒。且已知控制器模型为 $G_c(z)=0.3\dfrac{z-0.6}{z+0.8}$，试分析单位负反馈下闭环系统的稳定性。

求解 闭环系统的特征根及其模可以由下面的 MATLAB 语句求出

```
>> num=[6 -0.6 -0.12]; den=[1 -1 0.25 0.25 -0.125];
   H=tf(num,den,'Ts',0.1);    % 输入系统的传递函数模型
   z=tf('z','Ts',0.1); Gc=0.3*(z-0.6)/(z+0.8);  % 控制器模型
   GG=feedback(H*Gc,1);    % 闭环系统的模型
   pzmap(GG)   % 绘制系统的闭环零极点位置图
   abs(eig(GG)')  % 求取闭环系统的特征根的模，并将结果转置显示
```

可以得出各个特征根的模为 1.1644, 1.1644, 0.5536, 0.3232, 0.3232，可见，由于前两个特征根的模均大于 1，所以可以判定该闭环系统是不稳定的。闭环系统的零极点还可以由 pzmap(GG) 语句绘制出来，如图 3-2 所示。从图中可以看出，系统含有单位圆外的极点，所以系统是不稳定的。

利用系统零极点变换的语句 zpk(GG) 也能容易地得出系统的零极点模型

$$G(z)=\frac{1.8(z-0.6)(z-0.2)(z+0.1)}{(z-0.5536)(z^2-0.03727z+0.1045)(z^2+0.3908z+1.356)}$$

4. 线性系统的极点配置

令控制量将 $\boldsymbol{u}(t)=\boldsymbol{v}(t)-\boldsymbol{K}\boldsymbol{x}(t)$，代入开环系统的状态方程模型，则在状态反馈矩阵 $\boldsymbol{K}$ 下，系统的闭环状态方程模型可以写成

$$\begin{cases}\dot{\boldsymbol{x}}(t)=(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{v}(t)\\ \boldsymbol{y}(t)=(\boldsymbol{C}-\boldsymbol{D}\boldsymbol{K})\boldsymbol{x}(t)+\boldsymbol{D}\boldsymbol{v}(t)\end{cases}\tag{3-2-28}$$

如果一个系统是完全可控的，则可以通过引入状态反馈的方法将其闭环极点配置到任意预先指定的位置。假设闭环系统期望的极点位置为 $\mu_i, i=1,\cdots,n$，则闭环系统的特征方程 $\alpha(s)$ 可以表示成

$$\alpha(s)=\prod_{i=1}^{n}(s-\mu_i)=s^n+\alpha_1 s^{n-1}+\alpha_2 s^{n-2}+\cdots+\alpha_{n-1}s+\alpha_n\tag{3-2-29}$$

假设原系统的开环特征方程 $a(s)$ 可以写成

$$a(s)=\det(s\boldsymbol{I}-\boldsymbol{A})=s^n+a_1 s^{n-1}+a_2 s^{n-2}+\cdots+a_{n-1}s+a_n\tag{3-2-30}$$

若该系统完全可控，则状态反馈向量 $\boldsymbol{K}$ 可以由下式得出

$$\boldsymbol{K}=\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{\Gamma}^{-1}\boldsymbol{T}_{\mathrm{c}}^{-1}\tag{3-2-31}$$

其中 $\gamma^{\mathrm{T}} = [(\alpha_n - a_n), \cdots, (\alpha_1 - a_1)]$，$\boldsymbol{T}_{\mathrm{c}} = [\boldsymbol{B}, \boldsymbol{AB}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{B}]$ 为可控性判定矩阵，且

$$\boldsymbol{\Gamma} = \begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \\ a_{n-2} & a_{n-3} & \cdots & 1 & \\ \vdots & \vdots & \ddots & & \\ a_1 & 1 & & & \\ 1 & & & & \end{bmatrix} \tag{3-2-32}$$

可以看出因为 $\boldsymbol{\Gamma}$ 为非奇异 Hankel 矩阵，故该矩阵可逆。如果系统完全可控，则单变量系统的 $\boldsymbol{T}_{\mathrm{c}}$ 矩阵可逆，所以通过状态反馈向量 $\boldsymbol{K}$，可以任意地配置闭环系统的极点。基于此算法可以编写出 MATLAB 函数 bass_pp()，其清单在下面给出。

```
function K=bass_pp(A,B,p)
a1=poly(p); a=poly(A);  % 求出原系统和闭环系统的特征多项式
L=hankel(a(end-1:-1:1)); C=ctrb(A,B);
K=(a1(end:-1:2)-a(end:-1:2))*inv(L)*inv(C);
```

这里给出的极点配置方法只能用于单变量系统的极点配置，还有各种各样其他的极点配置方法，如 MATLAB 的控制系统工具箱提供的基于 Ackermann 算法的函数 acker() 和鲁棒极点配置算法 place()，它们的调用格式与前面的 bass_pp() 函数完全一致，其中 place() 函数还可以用于多变量系统的极点配置，在实际配置时应该使用此函数。

例 3-32 假设系统的状态方程模型为

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 2 & 0 & 0 & -2 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1 & 2 \\ 0 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{u}(t)$$

并想通过状态反馈将闭环系统的极点配置到 $-1, -2, -3, -4, -1 \pm \mathrm{j}$，则可以使用下面的语句直接进行极点配置，并检验闭环系统极点位置

```
>> A=[0,2,0,0,-2,0; 1,0,0,0,0,-1; 0,1,0,0,0,0;
      0,0,0,3,0,0;  2,0,0,1,0,0; 0,0,-1,0,1,0];
   B=[1,2; 0,0; 0,1; 0,-1; 0,1; 0,0];
   p=[-1 -2 -3 -4 -1+1i -1-1i]; % 期望闭环极点位置
   K=place(A,B,p),  % 系统极点配置
   eig(A-B*K)'  % 闭环系统极点检验，显示特征根向量的转置
```

可以设计出状态反馈矩阵 $\boldsymbol{K}$ 如下，且确实能将闭环系统极点配置到指定位置。

$$\boldsymbol{K}=\begin{bmatrix}7.9333 & -18.553 & -19.134 & 20.65 & 18.698 & 22.126\\ -0.36944 & -2.0412 & -2.3166 & -9.5475 & 0.57469 & 1.5013\end{bmatrix}$$

可以看出，由上面的语句可以立即设计出极点配置的状态反馈控制器矩阵，并将系统的闭环极点配置到预期的位置。注意，因为系统是多变量系统，所以 bass_pp() 和 acker() 函数均不能使用，只能使用 place() 函数进行极点配置。

5. 线性系统的频域响应分析

对连续线性系统 $G(s)$ 来说，如果用 $\mathrm{j}\omega$ 取代 s，则可以得出系统的频域响应模型$G(\mathrm{j}\omega)$。对一个已知的向量 $\boldsymbol{\omega}$ 做上述的替换，则得出的 $G(\mathrm{j}\boldsymbol{\omega})$ 为系统的频域响应向量。对系统频域分析来说，应该按照等对数间距构造向量 **ω=logspace(*a*,*b*,*N*)**，该语句将生成在 $(10^a,10^b)$ 区间上均匀分布的 $\boldsymbol{\omega}$ 向量，N 为频率点个数，其默认值为 50。

已知 $G(s)$ 求取 $G(\mathrm{j}\omega)$ 有两种常用方法，其一是用 $\mathrm{j}\boldsymbol{\omega}$ 向量分别取代 $G(s)$ 的分子和分母多项式中 s，则由得出的复数做除法就能得出所需的频域响应向量

```
ω=logspace(-3,3); ω1=ω*sqrt(-1); % 构造频率点向量
[num,den]=tfdata(G,'v'); % 提取传递函数模型
G1=polyval(num,ω1)./polyval(den,ω1);
```

另一种方法是基于状态方程的计算方法，由于系统状态方程由式 (3-2-27) 给出，故可以用循环的方式计算出每个频率点处 $G(\mathrm{j}\omega)$ 的值。

事实上，MATLAB 控制系统工具箱提供了强大的频域响应曲线绘制函数，直接使用下面的函数就可以绘制出各种频域响应曲线绘制函数

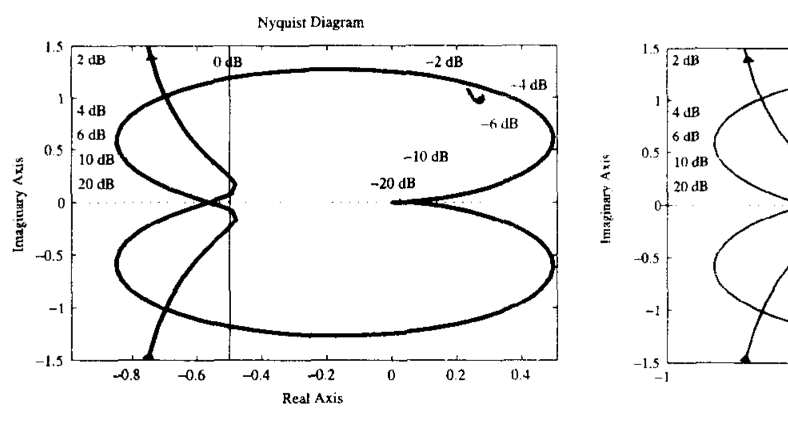
```
bode(G), bode(G,{ωm,ωM}), bode(G,ω),          % Bode 图
nyquist(G), nyquist(G,{ωm,ωM}), nyquist(G,ω), % Nyquist 图
nichols(G), nichols(G,{ωm,ωM}), nichols(G,ω), % Nichols 曲线
```

这些函数除了能精确地绘制出系统频域响应曲线外，还具有传统方法不具备的功能，如在 Nyquist 图上可以直接读取频率信息，弥补了传统频域响应的不足，为更好地进行系统的设计提供了强大的功能。另外，利用 grid 命令还可以在图上叠印系统的等 M 曲线和等 N 曲线。

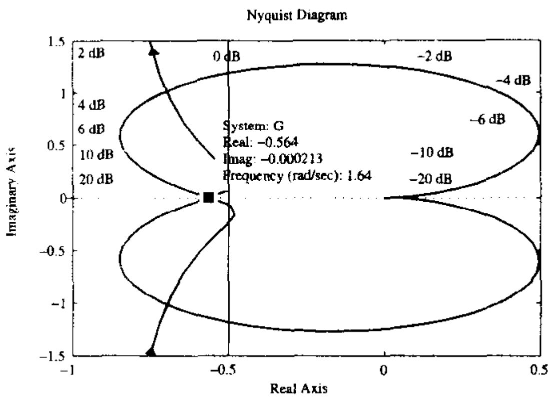
例 3-33 考虑连续线性系统的传递函数模型 $G(s)=\dfrac{s+8}{s(s^2+0.2s+4)(s+1)(s+3)}$，试绘制系统的 Nyquist 图，并读出该曲线和负实轴交点处的频率值。

求解 可以由下面的命令直接绘制出系统的 Nyquist 图，并叠印等幅值圆，如图 3-3 (a) 所示。由于传递函数分母含有 s，故当低频时增益很大，所以图中采用了 ylim() 函数对其 y 轴局部放大。

```
>> s=tf('s'); G=(s+8)/(s*(s^2+0.2*s+4)*(s+1)*(s+3));
   nyquist(G), grid   % 绘制 Nyquist 图并叠印等幅值圆
   ylim([-1.5 1.5])  % 根据需要手动选择纵坐标范围
```



(a) 系统的 Nyquist 图



(b) 从 Nyquist 图上读取附加信息

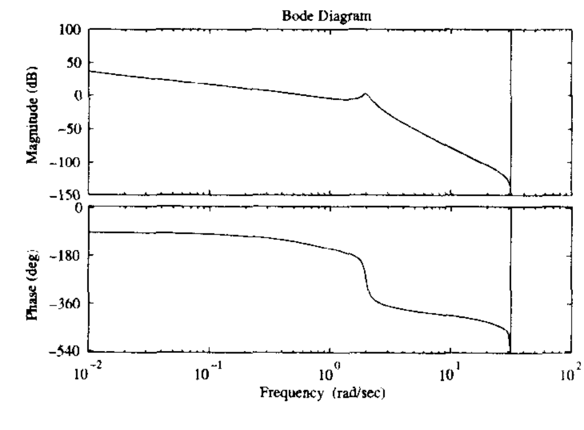
图 3-3 系统的频域响应分析结果

单击曲线上的任何一点则将显示出该点的频率信息，所以利用这里提供的新技术可以建立起增益的实部、虚部及频率直接的相互关系。单击 Nyquist 图和负实轴的交点，则可以得出如图 3-3 (b) 所示的信息显示，从而得出交点频率为 $\omega_c = 1.64\text{rad/s}$。

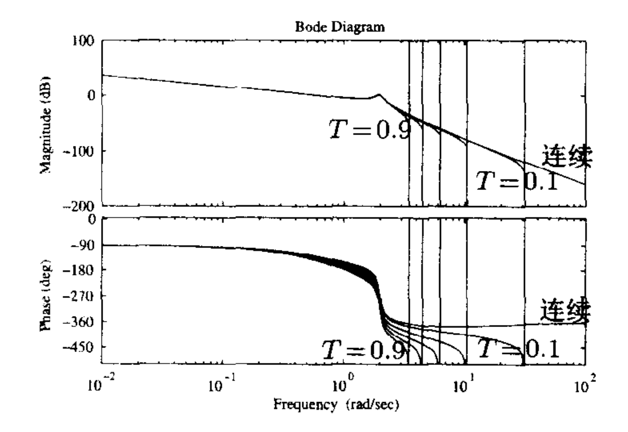
例 3-34 考虑前例的连续系统，试比较不同采样周期下离散化模型的 Bode 图。

求解 选择采样周期 $T = 0.1$ 秒，则可以得出离散化模型，该模型的 Bode 图可以用同样的命令直接绘制出来，如图 3-4 (a) 所示。

```
>> s=tf('s'); G=(s+8)/(s*(s^2+0.2*s+4)*(s+1)*(s+3));
   G1=c2d(G,0.1); bode(G1)
```



(a) $T = 0.1$ 时离散化系统的 Bode 图



(b) 不同采样周期下系统的 Bode 图

图 3-4 离散化系统的 Bode 图

选择不同的采样周期，则可以得出如图 3-4 (b) 所示的 Bode 图。随着采样周期的

不同选择，可以得出不同的 Bode 图。可见，低频时离散模型接近连续模型。采样周期越大，则高频响应与连续模型的差异越大。因为高频段对应于时域的初始响应，所以采样周期越大，开始时段系统的时域响应越不精确。

```
>> bode(G), hold on; for T=[0.1:0.2:1], bode(c2d(G,T)); end
```

6. 复数矩阵的对角占优与多变量系统的解耦

假设多变量反馈系统的前向通路传递函数矩阵为 $\boldsymbol{Q}(s)$，反馈通路的传递函数矩阵为 $\boldsymbol{H}(s)$，则闭环系统的传递函数矩阵为

$$\boldsymbol{G}(s)=[\boldsymbol{I}+\boldsymbol{Q}(s)\boldsymbol{H}(s)]^{-1}\boldsymbol{Q}(s) \tag{3-2-33}$$

其中 $\boldsymbol{I}+\boldsymbol{Q}(s)\boldsymbol{H}(s)$ 称为系统的回差 (return difference) 矩阵。因为稳定性分析利用回差矩阵的逆矩阵性质，所以在频域分析中用逆的 Nyquist 分析更方便，由此出现了在多变量频域分析系统中的逆 Nyquist 阵列 (inverse Nyquist array，INA) [13]方法。

类似于单变量系统，Nyquist 图是研究包围 $(-1,\mathrm{j}0)$ 点的周数来研究稳定性的，对回差矩阵的 INA 来说，可以研究其包围 $(0,\mathrm{j}0)$ 点的情形。

假设在某一频率 ω 下，多变量系统前向回路的 INA 表示为

$$\hat{\boldsymbol{Q}}(\mathrm{j}\omega)=\begin{bmatrix}\hat{q}_{11}(\mathrm{j}\omega) & \cdots & \hat{q}_{1p}(\mathrm{j}\omega)\\ \vdots & \ddots & \vdots\\ \hat{q}_{q1}(\mathrm{j}\omega) & \cdots & \hat{q}_{qp}(\mathrm{j}\omega)\end{bmatrix} \tag{3-2-34}$$

其中 $\hat{q}_{ij}(\mathrm{j}\omega)$ 为复数量。对于频率响应的所有数据来说，将由一系列 Gershgorin 圆的包络线可以构成 Gershgorin 带，若对全部的 ω 来说，各个对角元素的 Gershgorin 带均不包含原点，则称原系统为对角占优系统。显而易见，对角优势矩阵的特征根不位于原点处，则单位反馈的闭环系统是稳定的。

选定了频率向量 $\boldsymbol{w}$，并已知系统的多变量系统模型，则可以用多变量频域设计工具箱中提供的 `mv2fr()` 函数直接获得系统的频域响应数据

```
H=mv2fr(N,d,w)          % 已知多变量传递函数
H=mv2fr(A,B,C,D,w)      % 已知系统的状态方程模型
```

其中，返回的 $\boldsymbol{H}$ 是由多变量频率响应数据构成的矩阵，是多变量频域设计工具箱的基本数据格式。该工具箱提供了多变量系统的 Nyquist 图形绘制函数 `plotnyq()` 和 Gershgorin 圆绘制的函数 `fgersh()`，但由于调用过程较烦琐，所以对输入个数与输出个数相等的系统来说，编写了一个新的函数 **`inagersh(H)`**，可以直接绘制出系统带有 Gershgorin 带的逆 Nyquist 图，该函数的内容如下：

```
function inagersh(H,nij)
t=[0:.1:2*pi,2*pi]'; [nr,nc]=size(H); nw=nr/nc; ii0=1:nc;
```

```
if nargin==1, ii=1:nc; jj=1:nc;
else, ii=nij(1); jj=nij(2); end
for i=1:nc, circles{i}=[]; end
for k=1:nw  % 对各个频率获取逆 Nyquist 阵列
   Ginv=inv(H((k-1)*nc+1:k*nc,:)); nyq(:,:,k)=Ginv;
   for j=1:nc
      ij=find(ii0~=j);
      v=min([sum(abs(Ginv(ij,j))),sum(abs(Ginv(j,ij)))]);
      x0=real(Ginv(j,j)); y0=imag(Ginv(j,j));
      r=sum(abs(v)); % 计算 Gershgorin 圆的半径
      circles{j}=[circles{j}, x0+r*cos(t)+sqrt(-1)*(y0+r*sin(t))];
end,end
for i=ii, for j=jj   % 绘图
   if nargin==1, subplot(nc,nc,(i-1)*nc+j); end
   for k=1:nw, NN(k)=nyq(i,j,k); end
   if i==j,  % 对角图，带有 Gershgorin 带
      plot(real(NN),imag(NN),real(circles{i}),imag(circles{i}));
   else      % 非对角元素
      plot(real(NN),imag(NN))
   end,
end,end
```

如果该函数带两个输入变量，则第 2 个输入变量 n_{ij} 允许指定绘制某个 (i,j) 对的 Nyquist 图。和 MFD 工具箱提供的 plotnyq() 函数相比，inagersh() 函数首先是调用格式简单得多，另外，由于每个 Gershgorin 圆的半径选择行和列 Gershgorin 圆的最小半径，所以不再那么保守。

例 3-35 考虑多变量系统模型

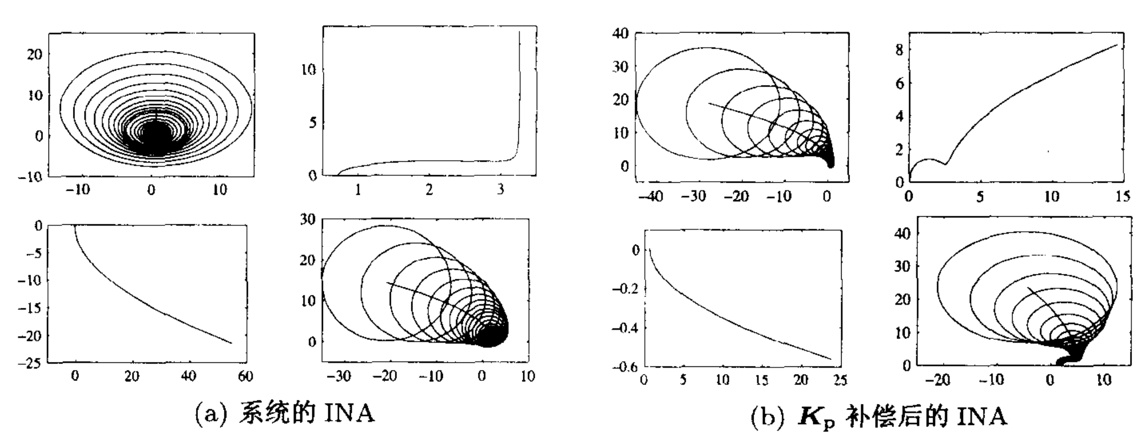
$$G(s)=\begin{bmatrix}\dfrac{0.806s+0.264}{s^2+1.15s+0.202} & \dfrac{-15s-1.42}{s^3+12.8s^2+13.6s+2.36}\\ \dfrac{1.95s^2+2.12s+0.49}{s^3+9.15s^2+9.39s+1.62} & \dfrac{7.15s^2+25.8s+9.35}{s^4+20.8s^3+116.4s^2+111.6s+18.8}\end{bmatrix}$$

试分析该系统是不是对角占优的系统，如果不是，试用前置补偿矩阵 $K_\mathrm{p}=\begin{bmatrix}0.3610 & 0.4500\\ -1.1300 & 1.0000\end{bmatrix}$ 对其补偿，分析其对角占优性。

求解 用下面的语句可绘制出系统的逆 Nyquist 曲线，如图 3-5 (a) 所示。

```
>> g11=tf([0.806 0.264],[1 1.15 0.202]);
   g12=tf([-15 -1.42],[1 12.8 13.6 2.36]);
   g21=tf([1.95 2.12 0.49],[1 9.15 9.39 1.62]);
   g22=tf([7.15 25.8 9.35],[1 20.8 116.4 111.6 18.8]);
   G=[g11, g12; g21, g22]; w=logspace(-2,1.5);
```

```
G=ss(G); H=mv2fr(G.a,G.b,G.c,G.d,w); inagersh(H); % INA 曲线
```



(a) 系统的 INA (b) K_p 补偿后的 INA

图 3-5 多变量系统的逆 Nyquist 阵列图

从图形可以看出，尽管闭环系统稳定，但由于 Gershgorin 带太宽，包围原点，不能保证为对角优势系统，所以在设计时有很多困难。

考虑前置补偿矩阵 K_p，则可以用下面的语句绘制出补偿系统的带有 Gershgorin 带的 INA 曲线，如图 3-5 (b) 所示。可见这时得出的 Gershgorin 带明显变窄，系统为对角优势系统，易于设计与进一步分析。

```
>> Kp=[0.3610,0.4500; -1.1300,1.0000];
   G=ss(G*Kp); H=mv2fr(G.a,G.b,G.c,G.d,w); inagersh(H); % INA 曲线
```

3.3 矩阵的基本变换与分解

本节首先引入矩阵的相似变换概念，然后给出矩阵的几种分解方法，如三角分解、Cholesky 分解、伴随矩阵变换、Jordan 变换和奇异值分析方法等，并介绍这些矩阵变换方法在控制系统研究中的应用。

3.3.1 矩阵的相似变换与正交矩阵

对某方阵 $\boldsymbol{A}$ 来说，如果存在一个非奇异的 $\boldsymbol{B}$ 矩阵，则可以通过下面的方式对原 $\boldsymbol{A}$ 矩阵进行变换。

$$\boldsymbol{X}=\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{B} \tag{3-3-1}$$

这样的变换称为相似变换，而 $\boldsymbol{B}$ 称为相似变换矩阵。相似变换后，$\boldsymbol{X}$ 矩阵的秩、迹、行列式和特征值等均不发生变化，其值和 $\boldsymbol{A}$ 矩阵完全一致。通过适当选择变换矩阵 $\boldsymbol{B}$，就能有目的地将任意给定的 $\boldsymbol{A}$ 矩阵相似变换成特殊的矩阵表示形式，而不改变原来 $\boldsymbol{A}$ 的重要性质。

对于一类特殊的相似变换矩阵 $\boldsymbol{T}$，如果它本身满足 $\boldsymbol{T}^{-1} = \boldsymbol{T}^*$，其中 $\boldsymbol{T}^*$ 为 $\boldsymbol{T}$ 的 Hermite 共轭转置矩阵，则称 $\boldsymbol{T}$ 为正交矩阵，并将之记为 $\boldsymbol{Q} = \boldsymbol{T}$。可见，正交矩阵 $\boldsymbol{Q}$ 满足条件

$$\boldsymbol{Q}^*\boldsymbol{Q} = \boldsymbol{I}, \text{ 且 } \boldsymbol{Q}\boldsymbol{Q}^* = \boldsymbol{I} \tag{3-3-2}$$

其中，$\boldsymbol{I}$ 为 $n \times n$ 的单位阵。

MATLAB 中提供了求取正交矩阵的函数 **Q=orth(A)**，该函数能求出 $\boldsymbol{A}$ 矩阵的正交基矩阵 $\boldsymbol{Q}$。若 $\boldsymbol{A}$ 为非奇异矩阵，则得出的正交基矩阵 $\boldsymbol{Q}$ 满足式 (3-3-2) 的条件。若 $\boldsymbol{A}$ 为奇异矩阵，则得出的矩阵 $\boldsymbol{Q}$ 的列数即为 $\boldsymbol{A}$ 矩阵的秩，且满足 $\boldsymbol{Q}^*\boldsymbol{Q} = \boldsymbol{I}$，而不满足 $\boldsymbol{Q}\boldsymbol{Q}^* = \boldsymbol{I}$。

例 3-36 求出 $\boldsymbol{A} = \begin{bmatrix} 5 & 9 & 8 & 3 \\ 0 & 3 & 2 & 4 \\ 2 & 3 & 5 & 9 \\ 3 & 4 & 5 & 8 \end{bmatrix}$ 矩阵的正交矩阵。

求解 一个已知矩阵的正交矩阵可以用 orth() 函数直接得出，并可以验证满足正交矩阵的性质。

```
>> A=[5,9,8,3; 0,3,2,4; 2,3,5,9; 3,4,5,8]; Q=orth(A),
   norm(Q'*Q-eye(size(A)), norm(eye(size(A)-Q*Q')
```

从而得出正交矩阵为 $\boldsymbol{Q} = \begin{bmatrix} -0.61967 & 0.77381 & -0.026187 & -0.12858 \\ -0.25485 & -0.15506 & 0.94903 & 0.10174 \\ -0.51978 & -0.52982 & -0.15628 & -0.65169 \\ -0.52999 & -0.31058 & -0.27245 & 0.74055 \end{bmatrix}$。

例 3-37 重新考虑例 3-9 中给出的奇异矩阵 $\boldsymbol{A}$，试求出其正交基矩阵。

求解 可以通过下面的 MATLAB 语句求取并检验其正交基矩阵。注意，因为 $\boldsymbol{A}$ 为奇异矩阵，故得出的 $\boldsymbol{Q}$ 为长方形矩阵。

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1]; Q=orth(A)
```

这时得出的正交基矩阵为 $\boldsymbol{Q} = \begin{bmatrix} -0.5 & 0.67082 & 0.5 \\ -0.5 & -0.22361 & -0.5 \\ -0.5 & 0.22361 & -0.5 \\ -0.5 & -0.67082 & 0.5 \end{bmatrix}$。

3.3.2 矩阵的三角分解和 Cholesky 分解

1. 一般矩阵的三角分解

矩阵的三角分解又称为 LU 分解，它的目的是将一个矩阵分解成一个下三角矩阵 $\boldsymbol{L}$ 和一个上三角矩阵 $\boldsymbol{U}$ 的乘积，亦即 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$，其中 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 矩阵可以分

别写成

$$\boldsymbol{L}=\begin{bmatrix}1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1\end{bmatrix},\quad \boldsymbol{U}=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn}\end{bmatrix} \tag{3-3-3}$$

由这两个矩阵可以简单地写出一个矩阵 $\boldsymbol{A}^F$, 其中

$$\boldsymbol{A}^F=\boldsymbol{L}+\boldsymbol{U}-\boldsymbol{I}=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & u_{nn}\end{bmatrix} \tag{3-3-4}$$

这样产生的矩阵与原来的 $\boldsymbol{A}$ 矩阵的关系可以写成

$$\begin{array}{llll}a_{11}=u_{11}, & a_{12}=u_{12}, & \cdots & a_{1n}=u_{1n} \\ a_{21}=l_{21}u_{11}, & a_{22}=l_{21}u_{12}+u_{22}, & \cdots & u_{2n}=l_{21}u_{1n}+u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}=l_{n1}u_{11}, & a_{n2}=l_{n1}u_{12}+l_{n2}u_{22}, & \cdots & a_{nn}=\sum\limits_{k=1}^{n-1}l_{nk}u_{kn}+u_{nn}\end{array} \tag{3-3-5}$$

由式 (3-3-5) 可以立即得出求取 l_{ij} 和 u_{ij} 的递推计算公式

$$l_{ij}=\frac{a_{ij}-\sum\limits_{k=1}^{j-1}l_{ik}u_{kj}}{u_{jj}},\ (j<i),\ \text{及}\ u_{ij}=a_{ij}-\sum_{k=1}^{i-1}l_{ik}u_{kj},\ (j\geqslant i) \tag{3-3-6}$$

该公式的递推初值为

$$u_{1i}=a_{1i},\ l_{i1}=a_{i1}/u_{11},\ \ i=1,2,\cdots,n \tag{3-3-7}$$

在 MATLAB 中也给出了基于主元素的矩阵 LU 分解函数 `lu()`，该函数的调用格式为

```
[L, U]=lu(A)        % LU 分解，A = LU
[L, U, P]=lu(A)     % P 为置换矩阵，A = P^-1 LU
```

其中，$\boldsymbol{L}, \boldsymbol{U}$ 分别为变换后的下三角和上三角矩阵。在 MATLAB 的 `lu()` 函数中考虑了主元素选取的问题，所以该函数一般会给出可靠的结果。由该函数得出的

下三角矩阵 $\boldsymbol{L}$ 并不一定是一个真正的下三角矩阵，因为选取它可能进行了一些元素行的交换，这样主对角线的元素可能不是 1，而在矩阵 $\boldsymbol{L}$ 内存在一个惟一的如式 (3-2-1) 中定义的置换，其各个元素的值均是 1。如果想获得有关换行信息，则可以由后一种格式调用 `lu()` 函数，这时 $\boldsymbol{P}$ 为单位阵变换出的置换矩阵，$\boldsymbol{A}$ 矩阵可以分解成 $\boldsymbol{A}=\boldsymbol{P}^{-1}\boldsymbol{L}\boldsymbol{U}$。

例 3-38 再考虑例 3-39 中矩阵的 LU 分解问题。分别用两种方法调用 MATLAB 中的 `lu()` 函数，则可以得出的不同结果。

求解 先输入 $\boldsymbol{A}$ 矩阵，并求出三角分解矩阵。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; [L1,U1]=lu(A)
```

这样可以将原矩阵分解为

$$\boldsymbol{L}_1=\begin{bmatrix}1 & 0 & 0 & 0\\ 0.3125 & 0.76852 & 1 & 0\\ 0.5625 & 0.43519 & 1 & 1\\ 0.25 & 1 & 0 & 0\end{bmatrix},\quad \boldsymbol{U}_1=\begin{bmatrix}16 & 2 & 3 & 13\\ 0 & 13.5 & 14.25 & -2.25\\ 0 & 0 & -1.8889 & 5.6667\\ 0 & 0 & 0 & 3.5527\times 10^{-15}\end{bmatrix}$$

可见，这样得出的 $\boldsymbol{L}_1$ 矩阵并非下三角矩阵，这是因为在分解过程中采用了主元素交换的方法。现在考虑 `lu()` 函数的另一种调用方法 `[`$\boldsymbol{L}$`,`$\boldsymbol{U}$`,`$\boldsymbol{P}$`]=lu(`$\boldsymbol{A}$`)`，得出

$$\boldsymbol{L}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0.25 & 1 & 0 & 0\\ 0.3125 & 0.76852 & 1 & 0\\ 0.5625 & 0.43519 & 1 & 1\end{bmatrix},\ \boldsymbol{U}=\begin{bmatrix}16 & 2 & 3 & 13\\ 0 & 13.5 & 14.25 & -2.25\\ 0 & 0 & -1.8889 & 5.6667\\ 0 & 0 & 0 & 3.5527\times 10^{-15}\end{bmatrix},\ \boldsymbol{P}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$$

注意，这里得出的 $\boldsymbol{P}$ 矩阵不是一个单位矩阵，而是单位矩阵的置换矩阵。结合得出的 $\boldsymbol{L}_1$ 矩阵可以看出，$\boldsymbol{P}$ 矩阵的 $p_{2,4}=1$，表明需要将 $\boldsymbol{L}_1$ 矩阵的第 4 行换到第 2 行，$p_{3,2}=p_{4,3}=1$ 表明需要将 $\boldsymbol{L}_1$ 的第 2 行换至第 3 行，将原第 3 行元素换至第 4 行，这样就可以得出一个真正的下三角矩阵 $\boldsymbol{L}$ 了，代入 `inv(`$\boldsymbol{P}$`)*`$\boldsymbol{L}$`*`$\boldsymbol{U}$，则可以精确地还原 $\boldsymbol{A}$ 矩阵。

由于前面给出的 `lu()` 函数并不能处理符号矩阵，所以可以依照该算法，先由式 (3-3-7) 求出 $\boldsymbol{U}$ 和 $\boldsymbol{L}$ 的第一行，然后由式 (3-3-6) 用递推的方法求出两个矩阵其他各行。利用 MATLAB 强大的矩阵运算功能，可以容易地设计出对符号矩阵的 LU 分解的函数 `lu()`，该函数应该置于 `@sym` 路径下，其内容为

```
function [L,U]=lu(A)
n=length(A); U=sym(zeros(size(A))); L=sym(eye(size(A)));
U(1,:)=A(1,:); L(:,1)=A(:,1)/U(1,1);
for i=2:n,
   for j=2:i-1, L(i,j)=(A(i,j)-L(i,1:j-1)*U(1:j-1,j))/U(j,j); end
   for j=i:n, U(i,j)=A(i,j)-L(i,1:i-1)*U(1:i-1,j); end
end
```

注意，在上述的算法中并未对主元素进行任何选取，因此该算法有时可能失效，因为在运算过程中 0 可能被用作除数。如果需要通用的分解程序，用户可以将主元素选取方法引入该函数[14]。

例 3-39 考虑例 3-9 中矩阵的 LU 分解问题。

求解 先输入 $\boldsymbol{A}$ 矩阵，并求出三角分解矩阵。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   [L,U]=lu(sym(A))
```

这样可以将原矩阵分解为

$$\boldsymbol{L}=\begin{bmatrix}1&0&0&0\\5/16&1&0&0\\9/16&47/83&1&0\\1/4&108/83&-3&1\end{bmatrix},\ \boldsymbol{U}=\begin{bmatrix}16&2&3&13\\0&83/8&145/16&63/16\\0&0&-68/83&204/83\\0&0&0&0\end{bmatrix}$$

2. 对称矩阵的三角分解 —— Cholesky 分解

如果 $\boldsymbol{A}$ 为对称矩阵，利用对称矩阵的特点则可以用 LU 分解的方法对之进行分解，将原来矩阵 $\boldsymbol{A}$ 分解成

$$\boldsymbol{A}=\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}=\begin{bmatrix}d_{11}&&&\\d_{21}&d_{22}&&\\\vdots&\vdots&\ddots&\\d_{n1}&d_{n2}&\cdots&d_{nn}\end{bmatrix}\begin{bmatrix}d_{11}&d_{21}&\cdots&d_{n1}\\&d_{22}&\cdots&d_{n2}\\&&\ddots&\vdots\\&&&d_{nn}\end{bmatrix}\tag{3-3-8}$$

如果利用对称矩阵的性质，则 LU 分解可以简化成

$$d_{ii}=\sqrt{a_{ii}-\sum_{k=1}^{i-1}d_{ik}^2},\ \ d_{ij}=\frac{a_{ij}-\sum\limits_{k=1}^{j-1}d_{ik}d_{jk}}{l_{jj}},\ (j<i)\tag{3-3-9}$$

该分解算法又称为对称矩阵是 Cholesky 分解算法。

MATLAB 提供了 `chol()` 函数来求取矩阵的 Cholesky 分解矩阵 $\boldsymbol{D}$，其结果为一个上三角矩阵。该函数的调用格式为 $\boldsymbol{D}$ = `chol(`$\boldsymbol{A}$`)` 。

例 3-40 考虑一个对称的 4 阶矩阵 $\boldsymbol{A}=\begin{bmatrix}9&3&4&2\\3&6&0&7\\4&0&6&0\\2&7&0&9\end{bmatrix}$，试得出其 Cholesky 分解。

求解 用下面的语句可以对之进行 Cholesky 分解，得出 $\boldsymbol{D}$ 矩阵。

```
>> A=[9,3,4,2; 3,6,0,7; 4,0,6,0; 2,7,0,9]; D=chol(A)
```

这样可以得出 $D=\begin{bmatrix}3 & 1 & 1.3333 & 0.66667\\0 & 2.2361 & -0.59628 & 2.8324\\0 & 0 & 1.9664 & 0.40684\\0 & 0 & 0 & 0.60648\end{bmatrix}$。

3. 正定、正规矩阵的定义与判定

正定矩阵是在对称矩阵基础上建立起来的概念。在介绍该概念之前，先给出主子行列式定义。假设对称矩阵 A 可以写成

$$A=\begin{bmatrix}a_{11} & a_{12} & a_{13} & \cdots & a_{1,n}\\a_{12} & a_{22} & a_{23} & \cdots & a_{2,n}\\a_{13} & a_{23} & a_{33} & \cdots & a_{3,n}\\\vdots & \vdots & \vdots & \ddots & \vdots\\a_{1,n} & a_{2,n} & a_{3,n} & \cdots & a_{n,n}\end{bmatrix} \tag{3-3-10}$$

则左上角的各个子矩阵的行列式称为主子行列式。如果一个对称矩阵所有的主子行列式均为正数，则称该矩阵为正定矩阵。

相应地，可以引入对称矩阵的半正定矩阵的概念，如果主子行列式均为非负的数值，则称为半正定矩阵。MATLAB 的函数 `chol()` 还可以用来判定矩阵的正定性。该函数的另一种调用格式为 `[D,p]=chol(A)`，式中对正定的 A 矩阵来说，返回的 $p=0$。所以可以利用这个性质来判定一个对称矩阵是否为正定矩阵。对非正定矩阵，则返回一个正的 p 值，$p-1$ 为 A 矩阵中正定的子矩阵的阶次，亦即 D 将为 $(p-1)$ 阶方阵。

如果复数方阵满足

$$A^*A=AA^* \tag{3-3-11}$$

则其中 A^* 为 A 的 Hermite 转置，亦即共轭转置，这样 A 矩阵称为正规矩阵。判定正规矩阵由定义可以直接完成 `norm(A'*A-A*A')`$<\epsilon$。如果得出的结果为 1，则 A 为正规矩阵。

例 3-41 试判定对称矩阵 $A=\begin{bmatrix}7 & 5 & 5 & 8\\5 & 6 & 9 & 7\\5 & 9 & 9 & 0\\8 & 7 & 0 & 1\end{bmatrix}$ 是否为正定矩阵，并求出其 Cholesky 分解矩阵。

求解 用下面的语句可以对 A 矩阵进行分解，得出 D 矩阵，并求出正定的阶次为 2，从而说明该矩阵并非正定矩阵，因为得出的 D 矩阵不满秩，且 $p\neq 0$。

```
>> A=[7,5,5,8; 5,6,9,7; 5,9,9,0; 8,7,0,1]; [D,p]=chol(A)
```

这样可以得出 $D=\begin{bmatrix}2.6458 & 1.8898\\0 & 1.5584\end{bmatrix}$，$p=3$。值得指出的是，非正定矩阵虽然不能进行 Cholesky 分解，但可以采用一般的 LU 分解

```
>> [L,U]=lu(sym(A))
```

这时本矩阵可以通过 LU 分解为

$$L=\begin{bmatrix}1&0&0&0\\5/7&1&0&0\\5/7&38/17&1&0\\8/7&9/17&73/57&1\end{bmatrix},\quad U=\begin{bmatrix}7&5&5&8\\0&17/7&38/7&9/7\\0&0&-114/17&-146/17\\0&0&0&124/57\end{bmatrix}$$

非对称矩阵也可以调用 chol() 函数，但结果是错误的，它首先将给定的矩阵强制按上三角子矩阵转换成对称矩阵。在严格的数学意义下，非对称矩阵是没有 Cholesky 分解的。

3.3.3　伴随矩阵变换

对一个给定的一般矩阵 $\boldsymbol{A}$，如果存在一个列向量 $\boldsymbol{v}$，使得如下生成的矩阵 $\boldsymbol{M}$ 为非奇异矩阵，则通过该矩阵可以将原矩阵 $\boldsymbol{A}$ 变换成伴随矩阵 $\boldsymbol{A}_\mathrm{c}$。

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{v}^\mathrm{T}\\\boldsymbol{v}^\mathrm{T}\boldsymbol{A}\\\vdots\\\boldsymbol{v}^\mathrm{T}\boldsymbol{A}^{n-1}\end{bmatrix}^{-1},\ \boldsymbol{A}_\mathrm{c}=\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{M}=\begin{bmatrix}0&1&0&\cdots&0\\0&0&1&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\-a_n&-a_{n-1}&-a_{n-2}&\cdots&-a_1\end{bmatrix}\tag{3-3-12}$$

其中，a_i 为 $\boldsymbol{A}$ 矩阵的特征多项式系数，$\det(s\boldsymbol{I}-\boldsymbol{A})=s^n+a_1s^{n-1}+\cdots+a_n$。可见，由于 $\boldsymbol{v}$ 矩阵可以任选，故将矩阵 $\boldsymbol{A}$ 变换成伴随矩阵的变换矩阵有无穷多种。

利用下面的 MATLAB 语句，则可以构造出变换矩阵 $\boldsymbol{M}^{-1}$。注意，因为本函数中引入了随机数向量，所以它得出的变换矩阵是不惟一的。

```
function M=comp_map(A)
n=length(A);
while 1, v=floor(0.5+rand(1,n)); M=v;
    for i=2:n, M(i,:)=M(i-1,:)*A; end
    if rank(M)==n, break; end,
end
```

例 3-42　给定矩阵 $\boldsymbol{A}=\begin{bmatrix}-10&-14&-17&-9\\10&16&18&10\\0&-1&0&0\\-9&-15&-17&-10\end{bmatrix}$，试选择变换矩阵 $\boldsymbol{M}$，将其变换成伴随矩阵。

求解　由前面的语句可以很容易地构造出变换矩阵

```
>> A=[-10,-14,-17,-9; 10,16,18,10; 0,-1,0,0; -9,-15,-17,-10];
   M1=sym(comp_map(A)), Ac=M1*A*inv(M1)
```

这样，可以得出变换矩阵和变换后的伴随矩阵为

$$M_1 = M^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ -10 & -15 & -17 & -9 \\ 31 & 52 & 53 & 30 \\ -60 & -105 & -101 & -59 \end{bmatrix}, \ A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & -6 & -7 & -4 \end{bmatrix}$$

3.3.4 矩阵的 Jordan 变换

对含有重特征值矩阵 A 通常不能直接分解成对角矩阵，而用纯特征值求解方法必定能使矩阵的特征向量矩阵 V 含有若干相同的列，使得该矩阵为奇异矩阵。

例 3-43 分别用数值方法和解析方法求出下面矩阵的特征值与特征向量矩阵。

$$A = \begin{bmatrix} -71 & -65 & -81 & -46 \\ 75 & 89 & 117 & 50 \\ 0 & 4 & 8 & 4 \\ -67 & -121 & -173 & -58 \end{bmatrix}$$

求解 用 MATLAB 语言中的数值算法可以求出该矩阵的特征值为

```
>> A=[-71,-65,-81,-46; 75,89,117,50; 0,4,8,4; -67,-121,-173,-58];
   D=eig(A)
```

这样可以得出矩阵的特征值为 $-8.0057, -8 \pm 0.0057323\mathrm{j}, -7.9943$。其实，该矩阵的特征值为位于 -8 的 4 重根，所以用数值解方法得出的特征值有很大的误差，故在这样的问题上不适合采用数值算法。如果采用解析算法 $[\boldsymbol{v},\boldsymbol{d}]$=eig(sym($\boldsymbol{A}$))，则可以得出如下结果:

$$v = \begin{bmatrix} 17/8 \\ -13/8 \\ 1 \\ -19/8 \end{bmatrix}, \ d = \begin{bmatrix} -8 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -8 \end{bmatrix}$$

可见，该矩阵特征值均为 -8，为 4 重根，故而得出的特征向量矩阵实际上是奇异矩阵，因为 4 列均相同，所以只保留了一列。

由于 MATLAB 语言、其他数值运算语言、软件包本身及数值算法的缺陷，均在计算内核中使用 double 型数据结构，导致对某些有重根矩阵的计算中间过程出现误差，所以在纯数值运算上将有很大的误差，即使使用现在最好的数值运算方法，如果不突破双精度的数据结构的限制也是难以避免的。

为解决这样的问题，需要使用符号运算工具箱中的 jordan() 函数来分解出

Jordan 标准型，并求出非奇异的变换矩阵。该函数的调用格式为

```
J=jordan(A)          % 只返回 Jordan 矩阵 J
[V,J]=jordan(A)      % 返回 Jordan 矩阵 J 和变换矩阵 V
```

有了变换矩阵 $\boldsymbol{V}$，则 Jordan 标准型可以由 $\boldsymbol{J}=\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}$ 变换出来。注意，Jordan 矩阵主对角线为矩阵的特征值，次主对角线为 1。

例 3-44 试对例 3-43 中给出的矩阵进行 Jordan 分解。

求解 符号矩阵的 Jordan 分解可用 jordan() 函数直接进行分解，得出所需的矩阵。

```
>> A=[-71,-65,-81,-46; 75,89,117,50; 0,4,8,4; -67,-121,-173,-58];
   [V,J]=jordan(sym(A))
```

利用符号运算工具箱可以立即得出

$$\boldsymbol{V}=\begin{bmatrix}-18496 & 2176 & -63 & 1\\ 14144 & -800 & 75 & 0\\ -8704 & 32 & 0 & 0\\ 20672 & -1504 & -67 & 0\end{bmatrix},\quad \boldsymbol{J}=\begin{bmatrix}-8 & 1 & 0 & 0\\ 0 & -8 & 1 & 0\\ 0 & 0 & -8 & 1\\ 0 & 0 & 0 & -8\end{bmatrix}$$

可见，这样得出的 $\boldsymbol{V}$ 矩阵就是满秩的矩阵了。对它求逆，就可以实现用普通数值运算难以实现的功能。该问题将在后面矩阵函数的例子中演示。

例 3-45 已知具有复数特征值的矩阵 $\boldsymbol{A}=\begin{bmatrix}-10 & -14 & -17 & -9\\ 10 & 16 & 18 & 10\\ 0 & -1 & 0 & 0\\ -9 & -15 & -17 & -10\end{bmatrix}$，试求出其 Jordan 分解。

求解 复数特征值矩阵的 Jordan 标准型及变换矩阵问题也可以由 jordan() 函数求取。可以用下面的语句得出相应的变换矩阵与 Jordan 矩阵为

```
>> A=[-10,-14,-17,-9; 10,16,18,10; 0,-1,0,0; -9,-15,-17,-10];
   [V,J]=jordan(sym(A))
```

这样可以得出 Jordan 变换矩阵为

$$\boldsymbol{V}=\begin{bmatrix}-11+2\mathrm{j} & -11-2\mathrm{j} & -5 & 23\\ 5-5\mathrm{j} & 5+5\mathrm{j} & 0 & -10\\ 5 & 5 & 0 & -10\\ -6+7\mathrm{j} & -6-7\mathrm{j} & 5 & 12\end{bmatrix},\quad \boldsymbol{J}=\begin{bmatrix}-1+\mathrm{j} & 0 & 0 & 0\\ 0 & -1-\mathrm{j} & 0 & 0\\ 0 & 0 & -1 & 1\\ 0 & 0 & 0 & -1\end{bmatrix}$$

在实际应用中，经常不希望使用复数型的 Jordan 块。如果某对共轭复数特征值为 $\lambda_{i,i+1}=\sigma\pm\mathrm{j}\omega$，期望的 Jordan 块可以写成 $\boldsymbol{J}_i=\begin{bmatrix}\sigma & \omega\\ -w & \sigma\end{bmatrix}$。若想获得这样的 Jordan 块，则需要对原来的变换矩阵做相应的处理，例如新变换矩阵的第 $i,i+1$ 列分别改写成原来这两列元素的实部和虚部即可。

例 3-46 重新考虑例 3-45 的 Jordan 标准型，试将其改变为实矩阵。

求解 分别令原来 V 矩阵的前两列为原来变换矩阵的实部和虚部，则可以直接对原矩阵解析变换

```
>> A=[-10,-14,-17,-9; 10,16,18,10; 0,-1,0,0; -9,-15,-17,-10];
   [V,J]=jordan(sym(A)); V(:,[1 2])=[real(V(:,1)), imag(V(:,1))]
   J=inv(V)*A*V
```

这时变换矩阵和 Jordan 矩阵分别为

$$V=\begin{bmatrix}-11 & 2 & -5 & 23\\ 5 & -5 & 0 & -10\\ 5 & 0 & 0 & -10\\ -6 & 7 & 5 & 12\end{bmatrix},\quad J=\begin{bmatrix}-1 & 1 & 0 & 0\\ -1 & -1 & 0 & 0\\ 0 & 0 & -1 & 1\\ 0 & 0 & 0 & -1\end{bmatrix}$$

利用底层算法也可以构造出 Jordan 变换矩阵和 Jordan 标准型。若 A_c 为伴随矩阵，并假设 Jordan 标准型可以最终写成

$$A=\operatorname{diag}(J_1,J_2,\cdots,J_{n_J}) \tag{3-3-13}$$

其中 n_J 为 Jordan 块的个数。若对应于 λ_i 特征值的特征向量为 v，则根据特征值的情况可以如下构造变换矩阵的相关列，并得出相应的变换矩阵：

① 若 A 矩阵的第 i 个特征值 λ_i 为 m_i 重的实根，则对应的变换子矩阵为

$$t_1=\begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ \lambda_i & 1 & 0 & \cdots & 0\\ \lambda_i^2 & 2\lambda_i & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \lambda_i^{n-1} & (n-1)\lambda_i^{n-2} & \dfrac{1}{2!}\dfrac{\mathrm{d}^2\lambda_i^{n-1}}{\mathrm{d}\lambda_i^2} & \cdots & \dfrac{1}{(n-1)!}\dfrac{\mathrm{d}^{m_i-1}\lambda_i^{n-1}}{\mathrm{d}\lambda_i^{m_i-1}}\end{bmatrix} \tag{3-3-14}$$

② 若第 i 和 $i+1$ 个根为一对共轭复根 $\lambda_i=\sigma_i\pm\mathrm{j}\omega_i$，则有 $t_i=\begin{bmatrix}\mathrm{Re}[v] & \mathrm{Im}[v]\end{bmatrix}$。

这样可以构造出由伴随矩阵到 Jordan 矩阵的变换矩阵 $T=[t_1,\cdots,t_{n_J}]$。由一般矩阵 A 变换成 Jordan 矩阵的变换矩阵可以由 $V=MT$ 求出，其中 M 由式 (3-3-12) 给出。

例 3-47 重新考虑例 3-42 中给出的矩阵，试将其变换成 Jordan 矩阵。

求解 例 3-42 已经得出了伴随矩阵及其变换矩阵。由下面语句

```
>> M1=sym(M1); Ac=M1*Ac*inv(M1); [V,D]=eig(Ac)
```

可以得出原矩阵的特征向量矩阵 V 和特征值矩阵 D 分别为

$$V=\begin{bmatrix}-1 & 1 & 1\\ 1 & -1+\mathrm{j} & -1-\mathrm{j}\\ -1 & -2\mathrm{j} & 2\mathrm{j}\\ 1 & 2+2\mathrm{j} & 2-2\mathrm{j}\end{bmatrix},\quad D=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1+\mathrm{j} & 0\\ 0 & 0 & 0 & -1-\mathrm{j}\end{bmatrix}$$

可以由下面语句进行变换

```
>> t1=(-1).^[0:3]'; T=[t1, [0; t1(1:3).*[1:3]']];
   T=[T real(V(:,3)) imag(V(:,3))], V=inv(M1)*T, J=inv(V)*A*V
```

从而得出各个变换矩阵和变换后的 Jordan 矩阵为

$$\boldsymbol{T}=\begin{bmatrix}1&0&1&0\\-1&1&-1&-1\\1&-2&0&2\\-1&3&2&-2\end{bmatrix},\ \boldsymbol{V}=\begin{bmatrix}1&-2&7/4&-1/4\\0&2&-1&-1/2\\0&2&-3/4&1/4\\-1&-5&5/4&3/4\end{bmatrix},\ \boldsymbol{J}=\begin{bmatrix}-1&1&0&0\\0&-1&0&0\\0&0&-1&-1\\0&0&1&-1\end{bmatrix}$$

3.3.5 矩阵的奇异值分解

矩阵的奇异值也是矩阵的一种测度。对任意的 $n\times m$ 矩阵 $\boldsymbol{A}$ 来说，总有

$$\boldsymbol{A}^{\rm T}\boldsymbol{A}\geqslant 0,\ \ \boldsymbol{A}\boldsymbol{A}^{\rm T}\geqslant 0 \tag{3-3-15}$$

且在理论上有

$$\operatorname{rank}(\boldsymbol{A}^{\rm T}\boldsymbol{A})=\operatorname{rank}(\boldsymbol{A}\boldsymbol{A}^{\rm T})=\operatorname{rank}(\boldsymbol{A}) \tag{3-3-16}$$

进一步可以证明，$\boldsymbol{A}^{\rm T}\boldsymbol{A}$ 与 $\boldsymbol{A}\boldsymbol{A}^{\rm T}$ 有相同的非负特征值 λ_i，在数学上把这些非负的特征值的平方根称作矩阵 $\boldsymbol{A}$ 的奇异值，记 $\sigma_i(\boldsymbol{A})=\sqrt{\lambda_i(\boldsymbol{A}^{\rm T}\boldsymbol{A})}$。

例 3-48 现在考虑一个演示例子[15]，假设矩阵 $\boldsymbol{A}=\begin{bmatrix}1&1\\\mu&0\\0&\mu\end{bmatrix}$，其中 $\mu=5\text{eps}$，试根据式 (3-3-16) 求取 $\boldsymbol{A}$ 矩阵的秩。

求解 显然，$\boldsymbol{A}$ 矩阵的秩为 2。用 MATLAB 运算也将得出同样的结论。

```
>> A=[1 1; 5*eps,0; 0,5*eps]; rank(A)
```

考虑式 (3-3-16) 中给出的方法计算矩阵 $\boldsymbol{A}$ 的秩。例如，用 $\boldsymbol{A}^{\rm T}\boldsymbol{A}$ 来求解 $\boldsymbol{A}$ 的秩，则可以得出

$$\boldsymbol{A}^{\rm T}\boldsymbol{A}=\begin{bmatrix}1+\mu^2&1\\1&1+\mu^2\end{bmatrix}$$

在双精度数值运算中，由于 μ^2 为 10^{-30} 级数值，所以加到 1 上事实上就已经被忽略了，所以 $\boldsymbol{A}^{\rm T}\boldsymbol{A}$ 矩阵将退化成幺矩阵，再求其秩显然为 1，从而可以断定原矩阵 $\boldsymbol{A}$ 的秩为 1，这与实际矛盾，故对这样的问题应该引入一个新的量作为矩阵的测度，即需要引入奇异值的概念。

假设 $\boldsymbol{A}$ 矩阵为 $n\times m$ 矩阵，则 $\boldsymbol{A}$ 矩阵可以分解为

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{M} \tag{3-3-17}$$

其中，L 和 M 为正交矩阵，$A_1=\mathrm{diag}(\sigma_1,\cdots,\sigma_n)$ 为对角矩阵，其对角元素满足不等式 $\sigma_1\geqslant\sigma_2\geqslant\cdots\geqslant\sigma_n\geqslant 0$。如果存在 $\sigma_n=0$，则原矩阵 A 为奇异矩阵，这时矩阵 A 的秩应该等于矩阵 A_1 中非 0 对角元素的个数。

MATLAB 提供了直接求取矩阵奇异值分解的函数 svd()，其调用方式为

```
S=svd(A)             % 只计算矩阵的奇异值
[L,A1,M]=svd(A)      % 矩阵奇异值与变换矩阵
```

其中，A 为原始矩阵，返回的 A_1 为对角矩阵，而 L 和 M 均为正交变换矩阵，并满足 $A=LA_1M^{\mathrm{T}}$。

矩阵的奇异值大小通常决定矩阵的性态，如果矩阵的奇异值变化特别大，则矩阵中某个元素有一个微小的变化将严重影响到原矩阵的参数，这样的矩阵又称为病态矩阵或坏条件矩阵，而在矩阵存在趋于 0 的奇异值时称为奇异矩阵。矩阵最大奇异值 $\sigma_{\max}$ 和最小奇异值 $\sigma_{\min}$ 的比值又称为该矩阵的条件数，记作 $\mathrm{cond}(A)$, 即 $\mathrm{cond}(A)=\sigma_{\max}/\sigma_{\min}$, 矩阵的条件数越大，则对元素变化越敏感。矩阵的最大奇异值和最小奇异值还分别记作 $\bar{\sigma}(A)$ 和 $\underline{\sigma}(A)$。在 MATLAB 中也提供了函数 cond(A) 来求取矩阵 A 的条件数。

例 3-49 试对例 3-9 中给出的 A 矩阵进行奇异值分解。

求解 如果调用 MATLAB 中给出的矩阵奇异值分解函数 svd(), 则可以容易地求出 L, A_1 和 M 矩阵以及该矩阵的条件数。

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1]; [L,A1,M]=svd(A)
```

则可以得出三个分解矩阵 L, A_1, M 为

$$\begin{bmatrix}-0.5&0.6708&0.5&-0.2236\\-0.5&-0.2236&-0.5&-0.6708\\-0.5&0.2236&-0.5&0.6708\\-0.5&-0.6708&0.5&0.2236\end{bmatrix},\begin{bmatrix}34&0&0&0\\0&17.889&0&0\\0&0&4.4721&0\\0&0&0&0\end{bmatrix},\begin{bmatrix}-0.5&0.5&0.6708&0.2236\\-0.5&-0.5&-0.2236&0.6708\\-0.5&-0.5&0.2236&-0.6708\\-0.5&0.5&-0.6708&-0.2236\end{bmatrix}$$

可见该矩阵含有 0 奇异值，故原矩阵为奇异矩阵。由 cond(A) 可以求出该矩阵的条件数为 3.2592×10^{16}，接近于 ∞，但在双精度数值运算上有一定的误差。

若先将 A 矩阵转换成符号矩阵，则调用 svd() 将得出更精确的奇异值分解矩阵。

例 3-50 对于 $n\neq m$ 的矩阵 $A=\begin{bmatrix}1&3&5&7\\2&4&6&8\end{bmatrix}$ 来说，也可以对之作奇异值分解。例如，可以对下面的长方形矩阵进行奇异值分解，并检验分解的结果。

求解 使用如下命令则可以得出

```
>> A=[1,3,5,7; 2,4,6,8]; [L,A1,M]=svd(A)
```

则可以得出三个分解矩阵 $\boldsymbol{L}$, $\boldsymbol{A}_1$, $\boldsymbol{M}$ 为

$$\begin{bmatrix}-0.6414 & -0.7672\\ -0.7672 & 0.6414\end{bmatrix}, \begin{bmatrix}14.269 & 0 & 0 & 0\\ 0 & 0.6268 & 0 & 0\end{bmatrix}, \begin{bmatrix}-0.1525 & 0.8227 & -0.3945 & -0.38\\ -0.3499 & 0.4214 & 0.2428 & 0.8007\\ -0.5474 & 0.0201 & 0.6979 & -0.4614\\ -0.7448 & -0.3812 & -0.5462 & 0.0407\end{bmatrix}$$

对这个例子进行逆运算，即 $\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{M}^{\mathrm{T}}$，则可以还原成原来的 $\boldsymbol{A}$ 矩阵。由前面的分析可见，这样得出的矩阵误差是很小的。

3.3.6 矩阵转换方法在控制理论研究中的应用举例

前面介绍的矩阵变换技术在控制理论研究中起着重要的作用。基于相似变换技术就可以将一个一般的状态方程变换成某种特殊的形式，以便于更好地进行系统的性质分析。本节首先给出系统的相似变换的概念，然后介绍基于矩阵相似变换的各种标准型及变换方法。

1. 线性系统的相似变换

假设存在一个非奇异矩阵 $\boldsymbol{T}$，且定义了一个新的状态变量向量 $\boldsymbol{z}$ 使得 $\boldsymbol{z}=\boldsymbol{T}^{-1}\boldsymbol{x}$，这样关于新状态变量 $\boldsymbol{z}$ 的状态方程模型可以写成

$$\begin{cases}\dot{\boldsymbol{z}}(t)=\boldsymbol{A}_{\mathrm{t}}\boldsymbol{z}(t)+\boldsymbol{B}_{\mathrm{t}}\boldsymbol{u}(t)\\ \boldsymbol{y}(t)=\boldsymbol{C}_{\mathrm{t}}\boldsymbol{z}(t)+\boldsymbol{D}_{\mathrm{t}}\boldsymbol{u}(t)\end{cases}, \text{且 } \boldsymbol{z}(0)=\boldsymbol{T}^{-1}\boldsymbol{x}(0) \tag{3-3-18}$$

式中 $\boldsymbol{A}_{\mathrm{t}}=\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}$, $\boldsymbol{B}_{\mathrm{t}}=\boldsymbol{T}^{-1}\boldsymbol{B}$, $\boldsymbol{C}_{\mathrm{t}}=\boldsymbol{C}\boldsymbol{T}$, $\boldsymbol{D}_{\mathrm{t}}=\boldsymbol{D}$。在矩阵 $\boldsymbol{T}$ 下的状态变换称为相似性变换，而 $\boldsymbol{T}$ 又称为相似变换矩阵。

控制系统工具箱中提供了 ss2ss()来完成状态方程模型的相似性变换，该函数的调用格式为 G_1=ss2ss(G,$\boldsymbol{T}$)，其中 G 为原始的状态方程模型，$\boldsymbol{T}$ 为变换矩阵，在 $\boldsymbol{T}$ 下的变换结果由 G_1 变量返回。注意，在本函数调用中输入和输出的变量都是状态方程对象，而不可以是其他对象。

2. 单变量控制系统的可控、可观测标准型转换

对单变量系统 (3-1-8) 来说，若系统的特征多项式可以写成

$$\det(s\boldsymbol{I}-\boldsymbol{A})=s^n+\alpha_1 s^{n-1}+\alpha_2 s^{n-2}+\cdots+\alpha_{n-1}s+\alpha_n \tag{3-3-19}$$

则可以构造出变换矩阵 $\boldsymbol{T}_{\mathrm{c}}$ [16]

$$\boldsymbol{T}_{\mathrm{c}}=\left[\boldsymbol{A}^{n-1}\boldsymbol{B},\boldsymbol{A}^{n-2}\boldsymbol{B},\cdots,\boldsymbol{A}\boldsymbol{B},\boldsymbol{B}\right]\begin{bmatrix}1 & & & \\ \alpha_1 & 1 & & \\ \vdots & \vdots & \ddots & \\ \alpha_{n-1} & \alpha_{n-2} & \cdots & 1\end{bmatrix} \tag{3-3-20}$$

这样就可以将原系统变换成可控标准型。由上面的算法，可以用 MATLAB 容易地写出变换矩阵

```
v=poly1(A);  % 求特征多项式系数，建议用 poly1() 取代 poly()
Tc=fliplr(ctrb(A,B))*flipud(hankel(v(end-1:-1:1)))
```

例 3-51 试求出下面系统的可控标准型。

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -4 & -3 & 0 & -1 \\ -3 & -7 & -1 & -3 \\ 0 & -1 & -13 & -1 \\ -1 & -3 & -1 & -10 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \\ -14 \\ 7 \\ 16 \end{bmatrix}u,\ y = \begin{bmatrix} 0 & 0 & 12 & 0 \end{bmatrix}\boldsymbol{x}$$

求解 由下面的语句可以得出转换矩阵，并得出可控标准型

```
>> A=[-4,-3,0,-1; -3,-7,-1,-3; 0,-1,-13,-1; -1,-3,-1,-10];
   B=[0; -14; 7; 16]; C=[0,0,12,0]; v=poly1(A);
   Tc=fliplr(ctrb(A,B))*flipud(hankel(v(end-1:-1:1)))
   Gc=ss(inv(Tc)*A*Tc,inv(Tc)*B,C*Tc,0)
```

变换矩阵和标准型分别为

$$\boldsymbol{T}_{\rm c} = \begin{bmatrix} 5445 & 776 & 26 & 0 \\ -9060 & -3909 & -433 & -14 \\ 1415 & 859 & 145 & 7 \\ 5400 & 3178 & 419 & 16 \end{bmatrix},\quad \left\{\begin{array}{l} \dot{\boldsymbol{z}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2105 & -1693 & -390 & -34 \end{bmatrix}\boldsymbol{z} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}u \\ y = \begin{bmatrix} 16980 & 10308 & 1740 & 84 \end{bmatrix}\boldsymbol{z} \end{array}\right.$$

可观测系统标准型是可控标准型的对偶形式。可观测标准型的变换矩阵为

$$\boldsymbol{T}_{\rm o}^{-1} = \begin{bmatrix} 1 & \cdots & \alpha_{n-2} & \alpha_{n-1} \\ & \ddots & \vdots & \vdots \\ & & 1 & \alpha_1 \\ & & & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{C}\boldsymbol{A}^{n-1} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C} \end{bmatrix} \tag{3-3-21}$$

类似于前面的可控标准型变换矩阵，可以由下面的语句定义出变换矩阵

```
v=poly1(A);  % 求特征多项式系数
To=inv(fliplr(hankel(v(end-1:-1:1)))*flipud(obsv(A,C)))
```

例 3-52 试求出例 3-51 中状态方程模型的可观测标准型。

求解 由下面语句可以直接求解本问题

```
>> A=[-4,-3,0,-1; -3,-7,-1,-3; 0,-1,-13,-1; -1,-3,-1,-10];
   B=[0; -14; 7; 16]; C=[0,0,12,0]; v=poly1(A);
```

```
To=inv(fliplr(hankel(v(end-1:-1:1)))*flipud(obsv(A,C)))
Go=ss(inv(To)*A*To,inv(To)*B,C*To,0)
```

变换矩阵和可观测标准型分别为

$$\boldsymbol{T}_{\rm o}=\begin{bmatrix}-1/20 & 1/3 & -13/6 & 169/12\\ -1/15 & 5/12 & -8/3 & 209/12\\ 0 & 0 & 0 & 1/12\\ 1/15 & -5/12 & 31/12 & -47/3\end{bmatrix},\quad \begin{cases}\dot{\boldsymbol{z}}=\begin{bmatrix}0 & 0 & 0 & -2105\\ 1 & 0 & 0 & -1693\\ 0 & 1 & 0 & -390\\ 0 & 0 & 1 & -34\end{bmatrix}\boldsymbol{z}+\begin{bmatrix}16980\\ 10308\\ 1740\\ 84\end{bmatrix}u\\ y=[0,0,0,1]u\end{cases}$$

3. 控制系统的 Jordan 标准型转换

系统的 Jordan 标准型可以由 `[Gj,Tj]=canon(G,'modal')` 函数直接求出。值得指出的是，若系统的 $\boldsymbol{A}$ 矩阵含有复数特征值，则用 `canon()` 函数不能得出正确的结果，应该结合前面 Jordan 变换的方法手工构造变换矩阵，得出合适的变换系统。

例 3-53 试求出例 3-51 中状态方程模型的 Jordan 标准型。

求解 由下面语句可以直接求解本问题

```
>> A=[-4,-3,0,-1; -3,-7,-1,-3; 0,-1,-13,-1; -1,-3,-1,-10];
   B=[0; -14; 7; 16]; C=[0,0,12,0];
   [Gj,Tj]=canon(ss(A,B,C,D),'modal')
```

得出的变换矩阵为

$$\boldsymbol{T}_{\rm j}=\begin{bmatrix}0.20668 & -0.14356 & 0.010657 & 0.02682\\ -0.12614 & -0.29795 & -0.59173 & -0.38757\\ -0.16674 & -0.28172 & 0.43726 & -0.39675\\ -0.12424 & -0.14959 & -0.0011401 & 0.15718\end{bmatrix}$$

在此变换矩阵下 Jordan 标准型为

$$\begin{cases}\dot{\boldsymbol{z}}=\begin{bmatrix}-2.0461 & 0 & 0 & 0\\ 0 & -14.159 & 0 & 0\\ 0 & 0 & -11.448 & 0\\ 0 & 0 & 0 & -6.3471\end{bmatrix}\boldsymbol{z}+\begin{bmatrix}2.5135\\ -6.172\\ 0.65693\\ 4.6012\end{bmatrix}\\ y=\begin{bmatrix}1.9932 & -11.736 & 11.512 & -0.21883\end{bmatrix}\boldsymbol{z}\end{cases}$$

4. 多变量系统的 Luenberger 可控标准型

多变量系统一种重要的可控标准型实现是 Luenberger 标准型，其具体实现方法是，构造可控性判定矩阵，并按照下面的顺序构成一个矩阵 $\boldsymbol{S}$[16]：

$$\boldsymbol{S}=\left(\boldsymbol{b}_1,\boldsymbol{A}\boldsymbol{b}_1,\cdots,\boldsymbol{A}^{\sigma_1-1}\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{A}^{\sigma_2-1}\boldsymbol{b}_2,\cdots,\boldsymbol{A}^{\sigma_p-1}\boldsymbol{b}_p\right)\qquad(3\text{-}3\text{-}22)$$

其中 σ_i 是能保证前面各列线性无关的最大指数值，亦即最大可控性指数，取该矩阵的前 n 列就可以构成一个 $n\times n$ 的方阵 $\boldsymbol{L}$。如果这样构成的满秩矩阵不足 n 列，亦即多变量系统不是完全可控，则可以在后面补足能够使得 $\boldsymbol{L}$ 为满秩方阵的列，可以通过添补随机数的方式构造该矩阵。该矩阵求逆，则可以按照如下的方式提取出相关各行

$$\boldsymbol{L}^{-1}=\begin{bmatrix} \boldsymbol{l}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{l}_{\sigma_1}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{l}_{\sigma_1+\sigma_2}^{\mathrm{T}} \\ \vdots \end{bmatrix} \begin{matrix} \\ \\ \leftarrow \text{提取此行} \\ \\ \leftarrow \text{提取此行} \\ \\ \end{matrix} \tag{3-3-23}$$

则可以依照下面的方法构造出变换矩阵逆阵 $\boldsymbol{T}^{-1}$

$$\boldsymbol{T}^{-1}=\begin{bmatrix} \boldsymbol{l}_{\sigma_1}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{l}_{\sigma_1}^{\mathrm{T}}\boldsymbol{A}^{\sigma_1-1} \\ \vdots \\ \boldsymbol{l}_{\sigma_1+\sigma_2}^{\mathrm{T}}\boldsymbol{A}^{\sigma_2-1} \\ \vdots \end{bmatrix} \tag{3-3-24}$$

通过变换矩阵 $\boldsymbol{T}$ 对原系统进行相似变换，即可以得出 Luenberger 标准型。前面介绍的方法很适合用 MATLAB 语言直接实现，根据算法，可以编写出如下的函数来生成变换矩阵 $\boldsymbol{T}$。

```
function T=luenberger(A,B)
n=size(A,1); p=size(B,2); S=[]; sigmas=[]; k=1;
for i=1:p
   for j=0:n-1
      S=[S,A^j*B(:,i)];
      if rank(S)==k, k=k+1;
      else, sigmas(i)=j-1; S=S(:,1:end-1); break; end,
   end
   if k>n, break; end
end
k=k-1; % 如果不是完全可控，则用随机数补足满秩矩阵
if k<n,
   while rank(S)~=n, S(:,k+1:n)=floor(10*rand(n,n-k)); end
end
L=inv(S); iT=[];
```

```
for i=1:p
   for j=0:sigmas(i), iT=[iT; L(i+sum(sigmas(1:i)),:)*A^j]; end
end
if k<n, iT(k+1:n,:)=L(k+1:end,:); end  % 不可控时补足满秩矩阵
T=inv(iT);  % 构造变换矩阵
```

例 3-54 试变换出下面给出的状态方程模型的 Luenberger 标准型。

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}15&6&-12&9\\4&14&8&-4\\2&4&10&-2\\9&6&-12&15\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}3&3\\2&2\\-2&-2\\3&9\end{bmatrix}\boldsymbol{u}(t)$$

求解 可以先输入系统的状态方程模型，用 luenberger() 函数建立起所需变换矩阵，最终获得系统的 Luenberger 标准型

```
>> A=[15,6,-12,9; 4,14,8,-4; 2,4,10,-2; 9,6,-12,15];
   B=[3,3; 2,2; -2,-2; 3,9]; T=luenberger(A,B) % Luenberger变换阵
   A1=inv(T)*A*T, B1=inv(T)*B  % 对系统进行变换，即可得出此标准型
```

其数学表示形式为

$$\boldsymbol{T}=\begin{bmatrix}18&3&61.2&3\\-48&2&-79.2&2\\48&-2&43.2&-2\\18&3&-46.8&9\end{bmatrix},\quad \dot{\boldsymbol{z}}(t)=\left[\begin{array}{cc:cc}0&1&0&0\\-144&30&-57.6&9.6\\\hdashline 0&0&0&1\\0&0&-108&24\end{array}\right]\boldsymbol{z}(t)+\left[\begin{array}{cc}0&0\\1&0\\\hdashline 0&0\\0&1\end{array}\right]\boldsymbol{u}(t)$$

这里不另行给出 Luenberger 可观测标准型的变换矩阵算法，因为可观测性问题与可控性问题是对偶的，故 $\boldsymbol{T}_{\rm o}$=inv(luenberger($\boldsymbol{A}$',$\boldsymbol{C}$')')。

5. 控制系统的结构分解

对于不完全可控的系统，还可以对之进行可控性阶梯分解，即构造一个状态变换矩阵 $\boldsymbol{T}$，就可以将系统的状态方程 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$ 变换成如下形式

$$\boldsymbol{A}_{\rm c}=\begin{bmatrix}\widehat{\boldsymbol{A}}_{\bar{\rm c}}&\boldsymbol{0}\\\widehat{\boldsymbol{A}}_{21}&\widehat{\boldsymbol{A}}_{\rm c}\end{bmatrix},\quad \boldsymbol{B}_{\rm c}=\begin{bmatrix}\boldsymbol{0}\\\widehat{\boldsymbol{B}}_{\rm c}\end{bmatrix},\quad \boldsymbol{C}_{\rm c}=[\widehat{\boldsymbol{C}}_{\bar{\rm c}},\widehat{\boldsymbol{C}}_{\rm c}] \tag{3-3-25}$$

该形式称为系统的可控阶梯分解形式，这样就可以将系统的不可控子空间 $(\widehat{\boldsymbol{A}}_{\bar{\rm c}},\boldsymbol{0},\widehat{\boldsymbol{C}}_{\bar{\rm c}})$ 和可控子空间 $(\widehat{\boldsymbol{A}}_{\rm c},\widehat{\boldsymbol{B}}_{\rm c},\widehat{\boldsymbol{C}}_{\rm c})$ 直接分离出来。构造这样的变换矩阵不是简单的事，但可以借用 MATLAB 中的现成函数 ctrbf() 对状态方程模型进行这样的阶梯分解 [$\boldsymbol{A}_{\rm c}$,$\boldsymbol{B}_{\rm c}$,$\boldsymbol{C}_{\rm c}$,$\boldsymbol{T}_{\rm c}$]=ctrbf($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$)，该函数就可以自动生成相似变换矩阵 $\boldsymbol{T}_{\rm c}$，将原系统模型直接变换成可控性阶梯分解模型。如果原来系统的状态方程模型是完全可控的，则此分解不必进行。

例 3-55 试求出例 3-28 中给出的不完全可控系统的可控阶梯形式。

求解 通过下面的语句对之进行分解，即可得出可控性阶梯分解形式

```
>> A=[-2.2,-0.7,1.5,-1; 0.2,-6.3,6,-1.5;...
       0.6,-0.9,-2,-0.5; 1.4,-0.1,-1,-3.5];
   B=[6,9; 4,6; 4,4; 8,4]; C=[1 2 3 4];
   [Ac,Bc,Cc,Tc]=ctrbf(A,B,C)
```

得出的可控阶梯标准型为

$$\hat{\boldsymbol{x}}[(k+1)T]=\left[\begin{array}{c:ccc} -4 & 0 & 0 & 0 \\ \hdashline -4.638 & -3.823 & -0.5145 & -0.127 \\ -3.637 & 0.1827 & -3.492 & -0.1215 \\ -4.114 & -1.888 & 1.275 & -2.685 \end{array}\right]\hat{\boldsymbol{x}}(kT)+\left[\begin{array}{cc} 0 & 0 \\ \hdashline 0 & 0 \\ 2.754 & -2.575 \\ -11.15 & -11.93 \end{array}\right]\boldsymbol{u}(kT)$$

系统的阶梯标准型变换矩阵是不惟一的，用前面介绍的 luenberger() 函数也可以构造出可控阶梯标准型的变换矩阵的逆矩阵。下面的语句仍然能构造出变换矩阵，得出系统的可控阶梯形式。

```
>> T=inv(luenberger(A,B)), Gc1=ss2ss(ss(A,B,C,0),T)
```

这样可以得出变换矩阵和阶梯标准型分别为

$$\boldsymbol{T}_{\rm o}=\begin{bmatrix} -0.5 & -0.75 & 2.75 & -0.5 \\ 2 & 6 & -18 & 4.5 \\ -7.7 & -23.45 & 70.5 & -17.75 \\ 2 & 7 & -20 & 5 \end{bmatrix},\quad \left\{\begin{array}{l} \boldsymbol{z}_{k+1}=\left[\begin{array}{ccc:c} -3 & 1 & 0 & -0.8 \\ 0 & 0 & 1 & 0 \\ 0 & -12 & -7 & -0.1 \\ \hdashline 0 & 0 & 0 & -4 \end{array}\right]\boldsymbol{z}_k+\left[\begin{array}{cc} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ \hdashline 0 & 0 \end{array}\right]\boldsymbol{u}_k \\ y_k=\left[\begin{array}{ccc:c} 58 & 231 & 49 & -27.35 \end{array}\right]\boldsymbol{z}_k \end{array}\right.$$

可观测性阶梯标准型可以由 obsvf() 函数直接求解，也可以由 ctrbf() 函数求解，因为可控性问题和可观测性问题是对偶的，所以可以由下面语句求解

$[\boldsymbol{A}_1,\boldsymbol{C}_{\rm o},\boldsymbol{B}_{\rm o},\boldsymbol{T}_1]$=ctrbf($\boldsymbol{A}'$,$\boldsymbol{C}'$,$\boldsymbol{B}'$)，且 $\boldsymbol{T}_{\rm o}$=inv($\boldsymbol{T}_1'$)，$\boldsymbol{A}_{\rm o}$=$\boldsymbol{A}_1'$

6. Lyapunov 定理与正定函数

100 多年前，俄国学者 Lyapunov 建立了一套非线性系统稳定性判定的方法。他通过构造 Lyapunov 代数函数，由其正定性来定性判定系统的稳定性。Lyapunov 定理是建立在标量函数 $V(\boldsymbol{x})$ 的正定、负定定义下的。若函数 $V(\boldsymbol{x})$ 在非零状态 $\neq \boldsymbol{0}$ 下值恒为正，即 $V(\boldsymbol{x})>0$，则称 $V(\boldsymbol{x})$ 为正定函数，若恒为负，则称负定函数。

例 3-56 试判定函数 $V(\boldsymbol{x})=10x_1^2+4x_2^2+x_3^2+2x_1x_2-2x_2x_3-4x_1x_3$ 的正定性质。

求解 可见该函数为二次型函数，可以建立起其二次型矩阵 $Q=\begin{bmatrix} 10 & 1 & -2 \\ 1 & 4 & -1 \\ -2 & -1 & 1 \end{bmatrix}$，这样就可以通过判定矩阵正定的方式判定其正定性质。由于得出的 Cholesky 矩阵为 3×3 的，所以该矩阵是正定的，也就是说，原二次型函数 $V(\boldsymbol{x})$ 是正定的。

```
>> Q=[10 1 -2; 1 4 -1; -2 -1 1]; chol(Q)
```

Lyapunov 定理为：对系统方程 $\dot{x}=\boldsymbol{f}(t,\boldsymbol{x})$ 及平衡点 $\boldsymbol{f}(t,\boldsymbol{0})\equiv 0$ 来说，若存在具有连续一阶可导的正定标量函数 $V(t,\boldsymbol{x})$，其一阶导数 $\dot{V}(t,\boldsymbol{x})$ 为负定的，则平衡点是一致渐近稳定的。当然，这种稳定性判定的难点在于 Lyapunov 函数的构造。

例 3-57 试判定系统 $\begin{cases}\dot{x}_1=-x_1(x_1^2+x_2^2)+x_2\\ \dot{x}_2=-x_1-x_2(x_1^2+x_2^2)\end{cases}$ 在原点处的稳定性。

求解 可以先构造出 Lyapunov 函数 $V(\boldsymbol{x})=x_1^2+x_2^2$，显然该函数是正定的。该函数的全导数可以写成 $\dot{V}(\boldsymbol{x})=\dfrac{\partial V}{\partial x_1}\dot{x}_1+\dfrac{\partial V}{\partial x_2}\dot{x}_2$。函数的导数可以由符号运算工具箱提供的 diff() 求出，故可以由下面语句求出 $\dot{V}(\boldsymbol{x})$ 为

```
>> syms x1 x2; V=x1^2+x2^2; f1=-x1*(x1^2+x2^2)+x2;
   f2=-x1-x2*(x1^2+x2^2); dV=diff(V,x1)*f1+diff(V,x2)*f2;
   simple(dV)
```

从而得出 $\dot{V}(\boldsymbol{x})=-2(x_1^2+x_2^2)^2<0$。故可以得出结论，原系统是渐近稳定的。

7. 均衡实现与模型降阶

在介绍均衡实现之前，先给出一个例子来演示均衡实现的必要性。

例 3-58 先考虑下面的系统状态方程模型[17]

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-1 & 0\\ 0 & -2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}10^{-6}\\ 10^{6}\end{bmatrix}u,\quad y(t)=[10^6\ \ 10^{-6}]\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

可见，$\boldsymbol{B}$ 向量中的两个元素与 $\boldsymbol{C}$ 向量中的相应元素有极大的差异，若用户选择新的状态变量 $z_1=10^6x_1$ 与 $z_2=10^{-6}x_2$，则可以将原来的系统相似地变换成

$$\begin{bmatrix}\dot{z}_1\\ \dot{z}_2\end{bmatrix}=\begin{bmatrix}-1 & 0\\ 0 & -2\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}+\begin{bmatrix}1\\ 1\end{bmatrix}u,\ \ y(t)=[1\ \ 1]\begin{bmatrix}z_1\\ z_2\end{bmatrix}$$

其中新的 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 向量中的相应元素则有了可比的尺度，从而使得系统的分析与设计更容易。因为这里引入的相似性变换重新定义了系统的坐标尺度，使得其更“均衡”，故这样的变换称为均衡变换。

MATLAB 的控制系统工具箱提供了 balreal()，可以由已知模型转换出均衡实现模型。该函数的调用格式为 [G_b,$\boldsymbol{g}$,$\boldsymbol{T}$]=balreal(G)，其中 G_b 为原系统均衡实现的状态方程模型，而 $\boldsymbol{g}$ 向量为从大到小排列的 Gram 矩阵元素，其大小反映出相应状态变量的重要程度。Gram 矩阵的详细定义在第 4 章给出。若原系统 G 由状态方程给出，则 $\boldsymbol{T}$ 矩阵为线性相似变换矩阵。

例 3-59 考虑下面的状态方程模型，试得出其均衡实现模型。

$$\begin{cases} \dot{x}(t) = \begin{bmatrix} -10 & -4.375 & -3.125 & -1.5 \\ 8 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0.5 & 0.4375 & 0.75 & 0.75 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

求解 从给出的状态方程并不能直接看出哪个状态在整个系统中更重要，所以应该给出下面语句来获得系统的均衡实现模型

```
>> A=[-10,-4.375,-3.125,-1.5; 8,0,0,0; 0,2,0,0; 0,0,1,0];
   B=[2; 0; 0; 0]; C=[0.5,0.4375,0.75,0.75]; G=ss(A,B,C,0)
   [G1,g,T]=balreal(G)
```

其均衡实现模型为

$$\begin{cases} \dot{z}(t) = \begin{bmatrix} -0.81996 & -0.31463 & 0.73015 & 0.07656 \\ 0.31463 & -0.44795 & 3.7879 & 0.23645 \\ 0.73015 & -3.7879 & -7.109 & -1.3934 \\ 0.07656 & -0.23645 & -1.3934 & -1.6231 \end{bmatrix} \boldsymbol{z}(t) + \begin{bmatrix} 0.92156 \\ -0.16627 \\ -0.42015 \\ -0.04307 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0.92156 & 0.16627 & -0.42015 & -0.04307 \end{bmatrix} \boldsymbol{z}(t) \end{cases}$$

可见，显然新状态 z_1 比 z_4 更重要。得出的 Gram 向量 $\boldsymbol{g}$ 和变换矩阵 $\boldsymbol{T}$ 为

$$\boldsymbol{g} = \begin{bmatrix} 0.51787 \\ 0.030858 \\ 0.012415 \\ 0.00057145 \end{bmatrix}, \quad \boldsymbol{T} = \begin{bmatrix} 0.46078 & 0.51264 & 0.83047 & 0.78661 \\ -0.083135 & -0.18125 & -0.040623 & 0.10147 \\ -0.21007 & 0.0092529 & 0.02343 & -0.021731 \\ -0.021535 & 0.020906 & -0.02947 & 0.021075 \end{bmatrix}$$

通过均衡实现，可以得出处理后系统的可控 Gram 矩阵，根据该矩阵的值可以看出哪些状态重要，哪些是次要的、对全局没有太大影响的，找到这些状态，则可以将其忽略掉，从而得出所需的降阶模型。

利用矩阵分块方法，可以重新写出原系统模型的均衡实现表示

$$\begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix} u, \quad y = [\boldsymbol{C}_1 \ \ \boldsymbol{C}_2] \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} + Du \tag{3-3-26}$$

假设状态向量 $\boldsymbol{x}_2$ 为快时变的模态，则在 t 很小时就可以近似地假设 $\dot{\boldsymbol{x}}_2 = \boldsymbol{0}$。这样可以由式 (3-3-26) 消去状态变量 $\boldsymbol{x}_2$，得出的状态方程模型为

$$\begin{cases} \dot{\boldsymbol{x}}_1 = \left(\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21}\right)\boldsymbol{x}_1 + \left(\boldsymbol{B}_1 - \boldsymbol{A}_{12}\boldsymbol{A}_{22}^{-1}\boldsymbol{B}_2\right)u \\ y = \left(\boldsymbol{C}_1 - \boldsymbol{C}_2\boldsymbol{A}_{22}^{-1}\boldsymbol{A}_{21}\right)\boldsymbol{x}_1 + \left(D - \boldsymbol{C}_2\boldsymbol{A}_{22}^{-1}\boldsymbol{B}_2\right)u \end{cases} \tag{3-3-27}$$

控制系统工具箱中给出了 modred() 函数来求取降阶模型，该函数的调用格式为 G_r=modred(G,elim)，其中 G 为均衡实现的原始模型，elim 为需要消去的状态变量，G_r 为降阶模型。

例 3-60 考虑例 3-59 中给出的系统模型，因为直接得出均衡后模型的 Gram 向量为 $\boldsymbol{g}^{\mathrm{T}}=[0.5179, 0.0309, 0.0124, 0.0006]$。显然，第 3, 4 个状态变量对输入、输出间的关联不是很重要，所以可以考虑消去这两个状态，得出降阶模型。调用模型降阶函数，则

```
>> A=[-10,-4.375,-3.125,-1.5; 8,0,0,0; 0,2,0,0; 0,0,1,0];
   B=[2; 0; 0; 0]; C=[0.5,0.4375,0.75,0.75]; G=ss(A,B,C,0)
   [G1,g,T]=balreal(G);  Gr=modred(G1,[3,4]);
```

从而得出降阶模型为

$$\dot{\boldsymbol{z}}=\begin{bmatrix}-0.74169 & -0.72862\\ 0.72862 & -2.6559\end{bmatrix}\boldsymbol{z}+\begin{bmatrix}0.87647\\ -0.40486\end{bmatrix}u,\ y=\begin{bmatrix}0.87647 & 0.40486\end{bmatrix}\boldsymbol{z}+0.025974u$$

3.4 矩阵方程的计算机求解

这里所谓的矩阵方程既包括线性矩阵方程，也包含矩阵的其他形式的方程，比如 Lyapunov 方程、Stein 方程、Sylvester 方程和 Riccati 方程，并介绍这些方程在控制系统研究中的应用。

3.4.1 线性方程组的计算机求解

考虑下面给出的线性代数方程

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{B} \tag{3-4-1}$$

其中，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 均为给定矩阵

$$\boldsymbol{A}=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ a_{m1} & a_{m2} & \cdots & a_{mn}\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}b_{11} & b_{12} & \cdots & b_{1p}\\ b_{21} & b_{22} & \cdots & b_{2p}\\ \vdots & \vdots & \ddots & \vdots\\ b_{m1} & b_{m2} & \cdots & b_{mp}\end{bmatrix} \tag{3-4-2}$$

可以由给定的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵构造出解的判定矩阵 $\boldsymbol{C}$

$$\boldsymbol{C}=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1p}\\ a_{21} & a_{22} & \cdots & a_{2n} & b_{21} & b_{22} & \cdots & b_{2p}\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_{m1} & b_{m2} & \cdots & b_{mp}\end{bmatrix} \tag{3-4-3}$$

这样可以不加证明地给出线性方程组有解的判定定理[18]：

① 当 $m=n$，且 $\text{rank}(\boldsymbol{A})=n$ 时，式 (3-4-1) 有惟一解

$$\boldsymbol{x}=\boldsymbol{A}^{-1}\boldsymbol{B} \tag{3-4-4}$$

用 MATLAB 语言可以立即得出该方程的解为 $\boldsymbol{x}$=inv($\boldsymbol{A}$)*$\boldsymbol{B}$。但是 inv() 函数的调用也有值得注意之处。例如，若 $\boldsymbol{A}$ 矩阵为接近奇异的矩阵，则利用此数值函数有可能产生错误的结果，故应该采用符号运算的 inv() 函数。

若采用符号运算工具箱，则可以直接使用 inv() 函数，如果能得出方程的解，则解是惟一的，如果出现错误信息，则再考虑其他的情形。

例 3-61 求解线性代数方程组 $\begin{bmatrix}1&2&3&4\\4&3&2&1\\1&3&2&4\\4&1&3&2\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}5&1\\4&2\\3&3\\2&4\end{bmatrix}$。

求解 上述方程可以用下面的语句直接求出，并验证其精度。

```
>> A=[1 2 3 4; 4 3 2 1; 1 3 2 4; 4 1 3 2]; B=[5 1; 4 2; 3 3; 2 4];
   x=inv(A)*B, norm(A*x-B)
```

则可以得出原方程的数值解为

$$\boldsymbol{x}=\begin{bmatrix}-1.8 & 2.400000000000001\\ 1.866666666666667 & -1.266666666666667\\ 3.866666666666668 & -3.266666666666668\\ -2.133333333333334 & 2.733333333333334\end{bmatrix}$$

将上面的解代入原方程，得出的误差为 7.4738×10^{-15}，可以看出误差是很小的，达到 10^{-15} 数量级。事实上，如果采用 $\boldsymbol{x}_1$=inv(sym($\boldsymbol{A}$))*$\boldsymbol{B}$ 符号运算语句将得出精确的解为 $\boldsymbol{x}_1=\begin{bmatrix}-9/5 & 12/5\\ 28/15 & -19/15\\ 58/15 & -49/15\\ -32/15 & 41/15\end{bmatrix}$，由 norm(double($\boldsymbol{A}$*$\boldsymbol{x}_1$-$\boldsymbol{B}$)) 可见误差为 0。

② 当 $\text{rank}(\boldsymbol{A})=\text{rank}(\boldsymbol{C})=r<n$ 时，式 (3-4-1) 有无穷多解，可以构造出线性方程组的 $n-r$ 个化零向量 $\boldsymbol{x}_i$，$i=1,2,\cdots,n-r$，原方程组对应的齐次方程组的解 $\hat{\boldsymbol{x}}$ 可以由 $\boldsymbol{x}_i$ 的线性组合来表示，即

$$\hat{\boldsymbol{x}}=\alpha_1\boldsymbol{x}_1+\alpha_2\boldsymbol{x}_2+\cdots+\alpha_{n-r}\boldsymbol{x}_{n-r} \tag{3-4-5}$$

其中，$\alpha_i, i=1,2,\cdots,n-r$ 为任意常数。在 MATLAB 语言中可以由 null() 直接求出，其调用格式为

$\boldsymbol{Z}$=null($\boldsymbol{A}$)	求解 $\boldsymbol{A}$ 矩阵的化零矩阵
$\boldsymbol{Z}$=null($\boldsymbol{A}$,'r')	求解 $\boldsymbol{A}$ 矩阵的化零矩阵的规范形式

null() 函数也可以用于符号变量描述方程的解析解问题，其中 Z 的列数为 $n-r$，而各列构成的向量又称为矩阵 A 的基础解系。

求解式 (3-4-1) 中给出的非齐次方程组也是较简单的，只要能求出该方程的任意一个特解 x_0，则原非齐次方程组的解为 $x=\hat{x}+x_0$。其实，在 MATLAB 语言中求解该方程的一个特解并非难事，用 x_0=pinv(A)*B 即可求出。

例 3-62 求解线性代数方程组 $\begin{bmatrix}1&2&3&4\\2&2&1&1\\2&4&6&8\\4&4&2&2\end{bmatrix}X=\begin{bmatrix}1\\3\\2\\6\end{bmatrix}$。

求解 用下面语句可以输入 A 和 B 矩阵，并按式 (3-4-3) 构造出 C 矩阵，从而判定矩阵方程的可解性。

```
>> A=[1 2 3 4; 2 2 1 1; 2 4 6 8; 4 4 2 2]; B=[1;3;2;6];
   C=[A B]; [rank(A), rank(C)]
```

通过检验秩的方法得出矩阵 A 和 C 的秩相同，都等于 2，小于矩阵的阶次 4，由此可以得出结论，原线性代数方程组有无穷多组解。可以考虑利用符号运算工具箱求解原问题，得出方程组的解析解。

```
>> Z=null(sym(A)), x0=sym(pinv(A))*B; syms a1 a2;
   x=a1*Z(:,1)+a2*Z(:,2)+x0 % 或更简单地，x1=x0+Z*[a1;a2]
```

则基础解系矩阵 Z 和特解向量 x_0，从而构造出通解向量 x。

$$Z=\begin{bmatrix}0&1\\1&0\\-6&-7\\4&5\end{bmatrix},\ x_0=\begin{bmatrix}\dfrac{125}{131}\\\dfrac{96}{131}\\\dfrac{125}{-10}\\\dfrac{-39}{131}\end{bmatrix},\ x=a_1\begin{bmatrix}0\\1\\-6\\4\end{bmatrix}+a_2\begin{bmatrix}1\\0\\-7\\5\end{bmatrix}+\begin{bmatrix}\dfrac{125}{131}\\\dfrac{96}{131}\\\dfrac{125}{-10}\\\dfrac{-39}{131}\end{bmatrix}=\begin{bmatrix}a_2+\dfrac{125}{131}\\a_1+\dfrac{96}{131}\\-6a_1-7a_2-\dfrac{10}{131}\\4a_1+5a_2-\dfrac{39}{131}\end{bmatrix}$$

其中 a_1, a_2 为任意常数。

③ 若 rank(A) < rank(C)，则式 (3-4-1) 为矛盾方程，这时只能利用 Moore-Penrose 广义逆求解出方程的最小二乘解为 x=pinv(A)*B，该解不满足原方程，只能使误差的范数测度 $||Ax-B||$ 取最小值。

例 3-63 如果 B 矩阵改成 $B=[1,2,3,4]^{\mathrm{T}}$，则通过求解可见

```
>> B=[1:4]'; C=[A B]; [rank(A), rank(C)]
```

这样，$\mathrm{rank}(A)=2\neq\mathrm{rank}(C)=3$，故原始方程是矛盾方程，不存在任何解。可以使用 pinv() 函数求取 Moore-Penrose 广义逆，从而求出原始方程的最小二乘解为

```
>> x=pinv(A)*B, A*x-B
```

方程的解和误差分别为

$$x=\begin{bmatrix}0.5465648855\\0.4549618321\\0.04427480916\\-0.04732824427\end{bmatrix}\text{，误差矩阵为}\begin{bmatrix}0.4\\8.8818\times10^{-16}\\-0.2\\1.7764\times10^{-15}\end{bmatrix}$$

显然，该解不满足原始代数方程组。

如果线性方程为

$$xA=B \tag{3-4-6}$$

则可以对上式两端进行转置处理，得出

$$A^{\mathrm{T}}z=B^{\mathrm{T}} \tag{3-4-7}$$

式中，$z=x^{\mathrm{T}}$，亦即可以得出形为式 (3-4-1) 的新线性方程，套用上述的几种情况则可以求解原始线性方程组。

3.4.2 Lyapunov 方程的计算机求解

1. 连续 Lyapunov 方程

连续 Lyapunov 方程可以表示成

$$AX+XA^{\mathrm{T}}=-C \tag{3-4-8}$$

Lyapunov 方程来源于微分方程稳定性理论，其中要求 $-C$ 为对称正定的 $n\times n$ 矩阵，从而可以证明解 X 亦为 $n\times n$ 对称矩阵。这类方程直接求解是很困难的，不过有了 MATLAB 这样的计算机数学语言，求解这样的问题就轻而易举了。可以由控制系统工具箱中提供的 `lyap()` 函数立即得出方程的解，该函数的调用格式为 `X=lyap(A,C)`，所以若给出 Lyapunov 方程中的 A 和 C，则可以立即获得相应 Lyapunov 方程的数值解。下面将通过例子演示一般 Lyapunov 方程的求解。

例 3-64 假设式 (3-4-8) 中的 A 和 C 矩阵分别为

$$A=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix},\quad C=-\begin{bmatrix}10&5&4\\5&6&7\\4&7&9\end{bmatrix}$$

试求解相应的 Lyapunov 方程，并验证解的精度。

求解 输入了给定的矩阵，可以由下面的 MATLAB 语句求出该方程的解。

```
>> A=[1 2 3;4 5 6; 7 8 0];  C=-[10, 5, 4; 5, 6, 7; 4, 7, 9];
   X=lyap(A,C), norm(A*X+X*A'+C)
```

可以得出原方程的数值解为

$$X=\begin{bmatrix}-3.944444444444445 & 3.88888888888889 & 0.388888888888891\\ 3.88888888888889 & -2.77777777777778 & 0.222222222222221\\ 0.388888888888891 & 0.222222222222221 & -0.111111111111112\end{bmatrix}$$

误差矩阵的范数为 2.0934×10^{-14}。可见，得出的方程解 X 基本满足原方程，且有较高精度。当然，后面还将给出解析解求解方法。

2. Lyapunov 方程的解析解

为方便叙述，可以将 Lyapunov 方程的各个矩阵参数表示为

$$X=\begin{bmatrix}x_1 & x_2 & \cdots & x_m\\ x_{m+1} & x_{m+2} & \cdots & x_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ x_{(n-1)m+1} & x_{(n-1)m+2} & \cdots & x_{nm}\end{bmatrix},C=\begin{bmatrix}c_1 & c_2 & \cdots & c_m\\ c_{m+1} & c_{m+2} & \cdots & c_{2m}\\ \vdots & \vdots & \ddots & \vdots\\ c_{(n-1)m+1} & c_{(n-1)m+2} & \cdots & c_{nm}\end{bmatrix}$$

利用 Kronecker 乘积的表示方法，可以将 Lyapunov 方程写成

$$(A\otimes I+I\otimes A)x=-c \tag{3-4-9}$$

可见，这样的方程有惟一解的条件并不局限于 $-C$ 为对称正定矩阵，形如式 (3-4-8) 的方程只要满足 $(A\otimes I+I\otimes A)$ 为非奇异的方阵即可保证惟一解。

例 3-65 仍考虑例 3-64 中给出的 Lyapunov 方程，试求出其解析解。

求解 由下面的语句可以求出其解析解，将其解代入原方程可以验证这一点。

```
>> A0=sym(kron(A,eye(3))+kron(eye(3),A));
   c=reshape(C',9,1);  x0=-inv(A0)*c; x=reshape(x0,3,3)'
```

可以得出方程的解析解为 $x=\begin{bmatrix}-71/18 & 35/9 & 7/18\\ 35/9 & -25/9 & 2/9\\ 7/18 & 2/9 & -1/9\end{bmatrix}$。

例 3-66 传统 Lyapunov 方程的条件 (C 为实对称正定矩阵) 能否突破？

求解 受微分方程稳定性影响，以前的传统观念似乎 Lyapunov 类方程有惟一解的充分必要条件是 $-C$ 矩阵为实对称正定矩阵。事实上，式 (3-4-12) 中给出的线性矩阵方程在不满足该条件的情况下仍有惟一解。例如，例 3-64 中给出的 A 矩阵不变，将 C 矩阵改为复数非对称矩阵

$$C=-\begin{bmatrix}1+1\mathrm{j} & 3+3\mathrm{j} & 12+10\mathrm{j}\\ 2+5\mathrm{j} & 6 & 11+6\mathrm{j}\\ 5+2\mathrm{j} & 11+\mathrm{j} & 2+12\mathrm{j}\end{bmatrix}$$

用上述方法可以输入 A 和 C 矩阵，可以立即解出满足该方程的复数解为

```
>> A=[1 2 3;4 5 6; 7 8 0];
   C=-[1+1i, 3+3i, 12+10i; 2+5i, 6, 11+6i; 5+2i, 11+1i, 2+12i];
   A0=sym(kron(A,eye(3))+kron(eye(3),A));
   c=reshape(C',9,1);  x0=-inv(A0)*c; x=reshape(x0,3,3)'
```

可以得出方程的解析解为

$$x = \begin{bmatrix} -5/102+1457/918\mathrm{j} & 15/17-371/459\mathrm{j} & -61/306+166/459\mathrm{j} \\ 4/17-626/459\mathrm{j} & -10/51+160/459\mathrm{j} & 115/153+607/459\mathrm{j} \\ -55/306+166/459\mathrm{j} & -26/153-209/459\mathrm{j} & 203/153+719/918\mathrm{j} \end{bmatrix}$$

经验证，得出的解确实满足原始的 Lyapunov 方程。故可以得出结论，如果不考虑 Lyapunov 方程稳定性的物理意义和 Lyapunov 函数为能量的物理原型，完全可以将 Lyapunov 方程进一步扩展成能处理任意 C 矩阵的情形。

3.4.3 Stein 方程的求解

Stein 方程的一般形式为

$$\boldsymbol{AXB}-\boldsymbol{X}+\boldsymbol{Q}=\boldsymbol{0} \tag{3-4-10}$$

这里，所有的矩阵均应该是 $n\times n$ 方阵。类似于前面的介绍，令 $\boldsymbol{X}$ 矩阵的向量展开为 $\boldsymbol{x}$，$\boldsymbol{Q}$ 矩阵的向量展开为 $\boldsymbol{q}$，则 Stein 方程可以由下面的线性方程直接解出。

$$\left(\boldsymbol{I}_{n\times n}-\boldsymbol{B}^{\mathrm{T}}\otimes\boldsymbol{A}\right)\boldsymbol{x}=\boldsymbol{q} \tag{3-4-11}$$

例 3-67 试求解 Stein 方程。

$$\begin{bmatrix} -2 & 2 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 2 \end{bmatrix}\boldsymbol{X}\begin{bmatrix} -2 & -1 & 2 \\ 1 & 3 & 0 \\ 3 & -2 & 2 \end{bmatrix}-\boldsymbol{X}+\begin{bmatrix} 0 & -1 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \end{bmatrix}=\boldsymbol{0}$$

求解 由下面的语句可以直接求解该方程

```
>> A=[-2,2,1; -1,0,-1; 1,-1,2]; B=[-2,-1,2; 1,3,0; 3,-2,2];
   Q=[0,-1,0; -1,1,0; 1,-1,-1]; x=inv(eye(9)-kron(B',A))*Q(:)
   X=reshape(x,3,3), norm(A*X*B-X+Q)
```

可以得出方程的解为 $\boldsymbol{X}=\begin{bmatrix} 0.087955 & 0.0081889 & -0.16998 \\ -0.02771 & 0.0094891 & -0.18375 \\ 0.21942 & 0.28077 & -0.045524 \end{bmatrix}$。代回原方程可得误差矩阵范数为 6.9774×10^{-16}。

作为 Stein 方程的一个特例，控制系统中采用的离散 Lyapunov 方程的一般表示形式为

$$\boldsymbol{AXA}^{\mathrm{T}}-\boldsymbol{X}+\boldsymbol{Q}=\boldsymbol{0} \tag{3-4-12}$$

在 Stein 方程中令 $\boldsymbol{B} = \boldsymbol{A}^{\mathrm{T}}$ 就可以得出离散 Lyapunov 方程。该方程可以由 MATLAB 控制系统工具箱的 `dlyap()` 函数直接求解。该函数的调用格式为 $\boldsymbol{X}$=`dlyap(`$\boldsymbol{A}$,$\boldsymbol{Q}$`)`，其实，如果 $\boldsymbol{A}$ 矩阵是非奇异矩阵，则等式两端同时右乘 $(\boldsymbol{A}^{\mathrm{T}})^{-1}$，就可以将其变换成连续的 Sylvester 方程，可以用第 3.4.4 节给出的算法求解其解析解。

例 3-68 求解下面的离散 Lyapunov 方程

$$\begin{bmatrix}8 & 1 & 6\\3 & 5 & 7\\4 & 9 & 2\end{bmatrix}\boldsymbol{X}\begin{bmatrix}8 & 1 & 6\\3 & 5 & 7\\4 & 9 & 2\end{bmatrix}^{\mathrm{T}} - \boldsymbol{X} + \begin{bmatrix}16 & 4 & 1\\9 & 3 & 1\\4 & 2 & 1\end{bmatrix} = \boldsymbol{0}$$

求解 该方程可以直接用 `dlyap()` 方程求解出来

```
>> A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
   X=dlyap(A,Q), norm(A*X*A'-X+Q)    % 精度验证
```

方程的解如下，且方程解的误差为 2.7778×10^{-15}。

$$\boldsymbol{X} = \begin{bmatrix}-0.164744254744667 & 0.0691496225433971 & -0.0167848097588811\\0.0528428990760884 & -0.0297849627662817 & -0.00615417767725561\\-0.1019783641208 & 0.0449589914281342 & -0.0305406582654483\end{bmatrix}$$

3.4.4 Sylvester 方程的计算机求解

Sylvester 方程的一般形式为

$$\boldsymbol{AX} + \boldsymbol{XB} = -\boldsymbol{C} \tag{3-4-13}$$

其中，$\boldsymbol{A}$ 为 $n\times n$ 矩阵，$\boldsymbol{B}$ 为 $m\times m$ 矩阵，$\boldsymbol{C}$ 和 $\boldsymbol{X}$ 均为 $n\times m$ 矩阵。该方程又称为广义的 Lyapunov 方程，式中 $\boldsymbol{A}$ 为 $n\times n$ 矩阵，$\boldsymbol{B}$ 为 $m\times m$ 矩阵。仍可以利用 MATLAB 控制系统工具箱中的 `lyap()` 函数直接求解该方程。该函数的一般调用格式为 $\boldsymbol{X}$=`lyap(`$\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$`)`，该函数采用的是 Schur 分解的数值解法求解方程。如果想得到解析解，类似于前述的一般 Lyapunov 方程，可以采用 Kronecker 乘积的形式将原始方程进行如下变换：

$$(\boldsymbol{A}\otimes\boldsymbol{I}_m + \boldsymbol{I}_n\otimes\boldsymbol{B}^{\mathrm{T}})\boldsymbol{x} = \boldsymbol{c} \tag{3-4-14}$$

如果 $(\boldsymbol{A}\otimes\boldsymbol{I}_m + \boldsymbol{I}_n\otimes\boldsymbol{B}^{\mathrm{T}})$ 矩阵为非奇异矩阵，则 Sylvester 方程有惟一解。

综合上述的算法，可以编写出 Sylvester 型方程的解析解求解程序 lyap.m，将其置于 `@sym` 目录下，以后再求解时只需将 $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ 矩阵之一设置成符号变量，就可以直接调用该函数了。这样改写的函数清单为

```
function X=lyap(A,B,C)  % 注意应该置于 @sym 目录下
```

```
if nargin==2, C=B; B=A'; end
[n,m]=size(C); A0=kron(A,eye(m))+kron(eye(n),B');
try
   C1=C'; x0=-inv(A0)*C1(:); X=reshape(x0,m,n)';
catch, error('singular matrix found.'), end
```

重新考虑 Stein 方程，如果方程右乘 $\boldsymbol{B}^{-1}$，则方程可以变换为

$$\boldsymbol{AX}+\boldsymbol{X}(-\boldsymbol{B})^{-1}=-\boldsymbol{Q}(\boldsymbol{B})^{-1} \tag{3-4-15}$$

故可以将该方程变换成 Sylvester 方程，由前面介绍的方法直接求解。类似地，考虑式 (3-4-12) 中给出的离散 Lyapunov 方程，两端同时右乘 $(\boldsymbol{A}^{\mathrm{T}})^{-1}$，则原来的离散 Lyapunov 方程可以变换成

$$\boldsymbol{AX}+\boldsymbol{X}[-(\boldsymbol{A}^{\mathrm{T}})^{-1}]=-\boldsymbol{Q}(\boldsymbol{A}^{\mathrm{T}})^{-1} \tag{3-4-16}$$

故令 $\boldsymbol{B}=-(\boldsymbol{A}^{\mathrm{T}})^{-1}$，$\boldsymbol{C}=\boldsymbol{Q}(\boldsymbol{A}^{\mathrm{T}})^{-1}$，则可以将其变换成式 (3-4-13) 所示的 Sylvester 方程，故也可以通过新的 `lyap()` 函数求解该方程。该函数的具体调用格式为

```
X=lyap(sym(A),C)                          % 连续 Lyapunov 方程
X=lyap(sym(A),-inv(B),Q*inv(B))           % Stein 方程
X=lyap(sym(A),-inv(A'),Q*inv(A'))         % 离散 Lyapunov 方程
X=lyap(sym(A),B,C)                        % Sylvester 方程
```

例 3-69 求解 Sylvester 方程 $\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}16&4&1\\9&3&1\\4&2&1\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$。

求解 调用 `lyap()` 函数可以立即得出原方程的数值解为

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[16,4,1; 9,3,1; 4,2,1];
   C=-[1,2,3; 4,5,6; 7,8,0]; X=lyap(A,B,C), norm(A*X+X*B+C)
```

可以得出方程的解为

$$\boldsymbol{X}=\begin{bmatrix}0.074871873700251&0.089913433762636&-0.432920003296282\\0.00807164473631289&0.481441768049986&-0.216033912855526\\0.0195770826298445&0.18264382872543&1.15792143917653\end{bmatrix}$$

其误差为 4.5315×10^{-15}。经检验可见该解精度较高。如果想获得原方程的解析解，则可以使用下面的语句，并可验证得出的解确实满足原始方程。

```
>> x=lyap(sym(A),B,C)
```

方程的解析解为

$$\boldsymbol{X}=\begin{bmatrix}1349214/18020305&648107/7208122&-15602701/36040610\\290907/36040610&3470291/7208122&-3892997/18020305\\70557/3604061&1316519/7208122&8346439/7208122\end{bmatrix}$$

例 3-70 重新考虑例 3-67 和例 3-68 中给出的 Stein 方程和离散 Lyapunov 方程，试求取其解析解。

求解 这两个方程可以通过下面的语句求解出解析解。

```
>> A=[-2,2,1; -1,0,-1; 1,-1,2]; B=[-2,-1,2; 1,3,0; 3,-2,2];
   Q=[0,-1,0; -1,1,0; 1,-1,-1]; X1=lyap(sym(A),-inv(B),Q*inv(B))
   A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
   X2=lyap(sym(A),-inv(A'),Q*inv(A')), norm(double(A*x*A'-x+Q))
```

方程的解为

$$\boldsymbol{X}_1=\begin{bmatrix}4147/47149 & 3861/471490 & -40071/235745\\ -2613/94298 & 2237/235745 & -43319/235745\\ 20691/94298 & 66191/235745 & -10732/235745\end{bmatrix}$$

$$\boldsymbol{X}_2=\begin{bmatrix}-22912341/139078240 & 48086039/695391200 & -11672009/695391200\\ 36746487/695391200 & -20712201/695391200 & -4279561/695391200\\ -70914857/695391200 & 31264087/695391200 & -4247541/139078240\end{bmatrix}$$

例 3-71 求解下面的 Sylvester 方程。

$$\boldsymbol{A}=\begin{bmatrix}8 & 1 & 6\\ 3 & 5 & 7\\ 4 & 9 & 2\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

求解 Sylvester 方程能解决的问题中并未要求 $\boldsymbol{C}$ 矩阵为方阵，利用上面的语句仍然能求出此方程的解析解，这里还可以尝试上面编写的 Lyapunov 方程解析解求解的新函数 lyap()，可以直接求解上述的方程。

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[2,3; 4,5]; C=-[1,2; 3,4; 5,6];
   X=lyap(sym(A),B,C)
```

得出方程的解为$\boldsymbol{X}=\begin{bmatrix}-2853/14186 & -11441/56744\\ -557/14186 & -8817/56744\\ 9119/14186 & 50879/56744\end{bmatrix}$。

3.4.5 Riccati 方程的计算机求解

Riccati 方程是一类很著名的二次型矩阵方程式，其一般形式为

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}-\boldsymbol{X}\boldsymbol{B}\boldsymbol{X}+\boldsymbol{C}=\boldsymbol{0} \tag{3-4-17}$$

由于含有未知矩阵 $\boldsymbol{X}$ 的二次项，所以 Riccati 方程的求解要比 Lyapunov 方程更难。当前不存在一般 Riccati 代数方程的解析解方法，只有各种数值方法，如基于 Schur 变换的算法[19] 和基于矩阵广义特征值的算法[20]。MATLAB 的控

制系统工具箱基于后者提供了现成函数 are()，可以直接求解式 (3-4-17) 中给出的方程，$\boldsymbol{X}$=are($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$)。值得指出的是，线性系统最优二次型控制中的 Riccati 方程实际上是这里能求解的 Riccati 方程的一个特例，因为 are() 函数还能求解 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 矩阵不对称，甚至是复数矩阵的 Riccati 方程。控制系统工具箱中 care() 函数甚至可以求解更难求解的 Riccati 方程[20]

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{E}+\boldsymbol{E}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{A}^{\mathrm{T}}+(\boldsymbol{E}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{B}+\boldsymbol{S})\boldsymbol{R}^{-1}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{E}+\boldsymbol{S}^{\mathrm{T}})+\boldsymbol{Q}=\boldsymbol{0} \tag{3-4-18}$$

这时，函数求解语句为 [$\boldsymbol{X}$,$\boldsymbol{L}$,$\boldsymbol{G}$]=care($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{Q}$,$\boldsymbol{R}$,$\boldsymbol{S}$,$\boldsymbol{E}$)。

例 3-72 考虑式 (3-4-17) 中给出的 Riccati 方程，其中

$$\boldsymbol{A}=\begin{bmatrix}-2&1&-3\\-1&0&-2\\0&-1&-2\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}2&2&-2\\-1&5&-2\\-1&1&2\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}5&-4&4\\1&0&4\\1&-1&5\end{bmatrix}$$

试求出该方程的数值解，并验证解的正确性。

求解 可以用下面的语句直接求解该方程。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
   C=[5 -4 4; 1 0 4; 1 -1 5]; X=are(A,B,C), norm(A'*X+X*A-X*B*X+C)
```

方程的解如下，解的误差为 1.8605×10^{-14}。

$$\boldsymbol{X}=\begin{bmatrix}0.9873949085&-0.7983276969&0.4188689966\\0.5774056496&-0.1307923365&0.5775477684\\-0.2840450002&-0.07303697833&0.6924114883\end{bmatrix}$$

3.4.6 矩阵方程求解在控制中的应用

矩阵方程求解在控制中有广泛的应用．例如，基于 Lyapunov 方程的求解可以得出控制系统的可控性、可观测性 Gram 矩阵，对系统的性质有更进一步的理解；系统范数的计算可以为求解鲁棒控制问题奠定基础；采用 Riccati 方程可以得出满足线性二次型指标的最优调节器。此外，线性系统辨识问题可以利用线性超定方程的最小二乘法求解，这部分内容将在第 6 章中另行介绍。

1. 控制系统的可控与可观测 Gram 矩阵

系统 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$ 的可控性和可观测性 Gram 矩阵分别定义如下

$$\boldsymbol{L}_{\mathrm{c}}=\int_0^{\infty}\mathrm{e}^{\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\mathrm{e}^{\boldsymbol{A}^{\mathrm{T}}t}\mathrm{d}t,\quad \boldsymbol{L}_{\mathrm{o}}=\int_0^{\infty}\mathrm{e}^{\boldsymbol{A}^{\mathrm{T}}t}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}\mathrm{e}^{\boldsymbol{A}t}\mathrm{d}t \tag{3-4-19}$$

系统的 Gram 矩阵实际上就是下面 Lyapunov 方程的解，所以由 lyap() 直接计算出这两个 Gram 矩阵

$$\boldsymbol{A}\boldsymbol{L}_{\mathrm{c}}+\boldsymbol{L}_{\mathrm{c}}\boldsymbol{A}^{\mathrm{T}}=-\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}},\quad \boldsymbol{A}^{\mathrm{T}}\boldsymbol{L}_{\mathrm{o}}+\boldsymbol{L}_{\mathrm{o}}\boldsymbol{A}=-\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C} \tag{3-4-20}$$

由 gram() 函数也可以计算 Gram 矩阵：gram(G,'c')，gram(G,'o')。

2. 控制系统的最小实现

对传递函数模型来说，如果其相同位置的零极点完全对消，则称为最小实现模型。最小实现的状态空间模型则是系统的既可控又可观测的子空间模型。

求取系统的最小实现可以按照以下的 3 个步骤。

① 求出可以将状态变量分解成可控、不可控子空间的相似性变换矩阵 $\boldsymbol{T}_{\rm c}^{-1}$

$$\boldsymbol{A}_{\rm c}=\boldsymbol{T}_{\rm c}^{-1}\boldsymbol{A}\boldsymbol{T}_{\rm c}=\begin{bmatrix}\widehat{\boldsymbol{A}}_{\bar{\rm c}} & 0\\ \widehat{\boldsymbol{A}}_{21} & \widehat{\boldsymbol{A}}_{\rm c}\end{bmatrix},\ \boldsymbol{B}_{\rm c}=\boldsymbol{T}_{\rm c}^{-1}\boldsymbol{B}=\begin{bmatrix}0\\ \widehat{\boldsymbol{B}}_{\rm c}\end{bmatrix},\ \boldsymbol{C}_{\rm c}=\boldsymbol{C}\boldsymbol{T}_{\rm c}=\begin{bmatrix}\widehat{\boldsymbol{C}}_{\bar{\rm c}} & \widehat{\boldsymbol{C}}_{\rm c}\end{bmatrix} \tag{3-4-21}$$

② 求出可以将可控子空间 $(\widehat{\boldsymbol{A}}_{\rm c},\widehat{\boldsymbol{B}}_{\rm c},\widehat{\boldsymbol{C}}_{\rm c})$ 分解成可观测子空间和不可观测子空间的相似性变换矩阵 $\widehat{\boldsymbol{T}}_{\rm o}$

$$\widehat{\boldsymbol{A}}_{\rm o}=\widehat{\boldsymbol{T}}_{\rm o}^{-1}\widehat{\boldsymbol{A}}_{\rm c}\widehat{\boldsymbol{T}}_{\rm o}=\begin{bmatrix}\widehat{\boldsymbol{A}}_{{\rm c},\bar{\rm o}} & \widehat{\boldsymbol{A}}_{{\rm c},12}\\ 0 & \widehat{\boldsymbol{A}}_{{\rm c,o}}\end{bmatrix},\ \widehat{\boldsymbol{B}}_{\rm o}=\widehat{\boldsymbol{T}}_{\rm o}^{-1}\widehat{\boldsymbol{B}}_{\rm c}=\begin{bmatrix}\widehat{\boldsymbol{B}}_{{\rm c},\bar{\rm o}}\\ \widehat{\boldsymbol{B}}_{\rm c,o}\end{bmatrix},\ \widehat{\boldsymbol{C}}_{\rm o}=\widehat{\boldsymbol{C}}_{\rm c}\widehat{\boldsymbol{T}}_{\rm o}=\begin{bmatrix}0 & \widehat{\boldsymbol{C}}_{\rm c,o}\end{bmatrix} \tag{3-4-22}$$

然后构造一个矩阵 $\widetilde{\boldsymbol{T}}_{\rm o}^{-1}=\begin{bmatrix}\boldsymbol{I}_{n-{\rm rank}\{\widehat{\boldsymbol{A}}_{\rm c}\}} & 0\\ 0 & \widehat{\boldsymbol{T}}_{\rm o}^{-1}\end{bmatrix}$。

③ 构造出相似性变换矩阵 $\boldsymbol{T}^{-1}=\widetilde{\boldsymbol{T}}_{\rm o}^{-1}\boldsymbol{T}_{\rm c}^{-1}$，这样得出的系统 $(\widehat{\boldsymbol{A}}_{\rm c,o},\widehat{\boldsymbol{B}}_{\rm c,o},\widehat{\boldsymbol{C}}_{\rm c,o})$ 即为原系统的最小实现形式。在这样的相似变换矩阵 $\boldsymbol{T}$ 下，整个系统的规范形式可以写成

$$\dot{\boldsymbol{z}}=\left[\begin{array}{c:cc}\widehat{\boldsymbol{A}}_{\bar{\rm c}} & \multicolumn{2}{c}{0}\\ \hdashline \widehat{\boldsymbol{A}}_{21} & \widehat{\boldsymbol{A}}_{{\rm c},\bar{\rm o}} & \widehat{\boldsymbol{A}}_{{\rm c},12}\\ & 0 & \widehat{\boldsymbol{A}}_{\rm c,o}\end{array}\right]\boldsymbol{z}+\left[\begin{array}{c}0\\ \hdashline \widehat{\boldsymbol{B}}_{{\rm c},\bar{\rm o}}\\ \widehat{\boldsymbol{B}}_{\rm c,o}\end{array}\right]\boldsymbol{u},\ \boldsymbol{y}=\left[\begin{array}{c:cc}\boldsymbol{C}_{\rm c} & 0 & \widehat{\boldsymbol{C}}_{\rm c,o}\end{array}\right]\boldsymbol{z}+\boldsymbol{D}\boldsymbol{u} \tag{3-4-23}$$

例 3-73 求出下面状态方程模型的最小实现。

$$\dot{\boldsymbol{x}}=\begin{bmatrix}-5 & 8 & 0 & 0\\ -4 & 7 & 0 & 0\\ 0 & 0 & 0 & 4\\ 0 & 0 & -2 & 6\end{bmatrix}\boldsymbol{x}+\begin{bmatrix}4\\ -2\\ 2\\ 1\end{bmatrix}u,\ y=[2\ \ -2\ \ -2\ \ 2]\boldsymbol{x}$$

求解 可以由下面的 MATLAB 语句实现上面 3 个步骤，得出系统的最小实现模型

```
>> A=[-5,8,0,0; -4,7,0,0; 0,0,0,4; 0,0,-2,6];
   B=[4;-2;2;1]; C=[2,-2,-2,2]; D=0; [Ac,Bc,Cc,Tc]=ctrbf(A,B,C);
   Ac1=Ac(2:4,2:4); Bc1=Bc(2:4); Cc1=Cc(2:4);
   [Ao,Bo,Co,To]=obsvf(Ac1,Bc1,Cc1); Ar=Ao(2:3,2:3); Br=Bo(2:3);
   Cr=Co(2:3); Gr=ss(Ar,Br,Cr,D), Gr1=zpk(Gr)
```

这样可以得出最小实现及其零极点模型为

$$\begin{cases} \dot{\boldsymbol{x}} = \begin{bmatrix} 1.7273 & 0.86244 \\ 0.86244 & -0.72727 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 3.4112 \\ -3.371 \end{bmatrix} u \\ y = [0, -2.9665]\boldsymbol{x} \end{cases}, \quad G_{\mathrm{r1}}(s) = \frac{10(s-2.6)}{(s-2)(s+1)}$$

3. 线性系统的范数求解

我们知道，矩阵范数可以用一个值来描述一个矩阵的“大小”。线性系统也有自己的范数，系统的范数是系统的一种测度：

① $\mathcal{H}_2$ 范数的定义为

$$||\boldsymbol{G}(s)||_2 = \sqrt{\frac{1}{2\pi \mathrm{j}} \int_{-\mathrm{j}\infty}^{\mathrm{j}\infty} |\boldsymbol{G}(\mathrm{j}\omega)|^2 \mathrm{d}\omega} \tag{3-4-24}$$

$\mathcal{H}_2$ 范数实际上是当输入信号为脉冲时，输出信号的平方积分的平方根，对随机信号来说，$\mathcal{H}_2$ 范数是在白噪声信号激励下输出信号的均方根。

定义矩阵 $\boldsymbol{L}$ 为系统的可控 Gram 矩阵，系统的 $\mathcal{H}_2$ 范数可以由下式直接求出 $||\boldsymbol{G}||_2 = \mathrm{tr}(\boldsymbol{CLC}^{\mathrm{T}})$。采用底层 MATLAB 语句，则由下面语句求解 `L=gram(G,'c'); g=trace(C*L*C')` 。

② $\mathcal{H}_\infty$ 范数的定义为

$$||\boldsymbol{G}(s)||_\infty = \sup_{u(t)\neq 0} \frac{|y(t)|_2}{|u(t)|_2} \tag{3-4-25}$$

式中 $u(t)$ 和 $y(t)$ 分别为系统的输入和输出信号，若系统稳定，则系统的 $\mathcal{H}_\infty$ 范数可以由下式求出

$$||\boldsymbol{G}(s)||_\infty = \sup_{\omega} |\boldsymbol{G}(\mathrm{j}\omega)| \tag{3-4-26}$$

从式子中可以看出，$\mathcal{H}_\infty$ 范数实际上是频域响应幅值的峰值。系统的 $\mathcal{H}_\infty$ 范数没有直接的解析方法，只能通过数值方法求解。对给定一个正数 $\gamma > 0$，当且仅当 Hamilton 矩阵

$$\boldsymbol{H}_\gamma = \begin{bmatrix} \boldsymbol{A} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{C} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}} \\ \boldsymbol{C}^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{D}\boldsymbol{R}^{-1}\boldsymbol{D}^{\mathrm{T}})\boldsymbol{C} & -(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{C})^{\mathrm{T}} \end{bmatrix} \tag{3-4-27}$$

没有纯虚数的特征值，不等式 $||\boldsymbol{G}||_\infty < \gamma$ 成立，其中 $\boldsymbol{R} = \gamma^2 \boldsymbol{I} - \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} > 0$。我们可以使用数值最优化算法来求解 γ 参数最大值，比如简单的二分算法。

利用 MATLAB 语言的控制系统工具箱的 norm() 函数，可以更容易地求出系统 $\boldsymbol{G}$ 的 $\mathcal{H}_2$ 和 $\mathcal{H}_\infty$ 范数：norm($\boldsymbol{G}$)，norm($\boldsymbol{G}$,'inf')。

4. 线性二次型最优控制器与调节器设计

假设线性系统的状态方程模型为

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{cases} \tag{3-4-28}$$

可以引入最优控制的性能指标，即设计一个输入量 $\boldsymbol{u}(t)$，使得

$$J = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(t_{\mathrm{f}})\boldsymbol{S}\boldsymbol{x}(t_{\mathrm{f}}) + \frac{1}{2}\int_{t_0}^{t_{\mathrm{f}}}\Big[\boldsymbol{x}^{\mathrm{T}}(t)\boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}^{\mathrm{T}}(t)\boldsymbol{R}(t)\boldsymbol{u}(t)\Big]\mathrm{d}t \tag{3-4-29}$$

为最小，其中 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 分别为对状态变量和输入变量的加权矩阵，t_{f} 为控制作用的终止时间。矩阵 $\boldsymbol{S}$ 对控制系统的终值也给出某种约束，这样的控制问题称为线性二次型 (linear quadratic，LQ) 最优控制问题。

由线性二次型最优控制理论[21]可知，若想最小化 J，则控制信号应该为

$$\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}(t)\boldsymbol{x}(t) \tag{3-4-30}$$

其中，$\boldsymbol{P}(t)$ 为对称矩阵，该矩阵满足下面著名的 Riccati 微分方程

$$\dot{\boldsymbol{P}}(t) = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}(t) - \boldsymbol{P}(t)\boldsymbol{A} + \boldsymbol{P}(t)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}(t) - \boldsymbol{Q} \tag{3-4-31}$$

$\boldsymbol{P}(t)$ 矩阵的终值为 $\boldsymbol{P}(t_{\mathrm{f}}) = \boldsymbol{S}$。可见，最优控制信号将取决于状态变量 $\boldsymbol{x}(t)$ 与 Riccati 微分方程的解 $\boldsymbol{P}(t)$。

可以看出，Riccati 微分方程求解是很困难的，而基于该方程解的控制器的实现就更困难，所以这里只考虑稳态问题这样的简单情况。在稳态的情况下，终止时间假定为 $t_{\mathrm{f}} \to \infty$，这样会使得系统的状态渐近地趋于 $\boldsymbol{0}$。Riccati 微分方程的解矩阵 $\boldsymbol{P}(t)$ 将趋于常数矩阵，使得 $\dot{\boldsymbol{P}}(t) = \boldsymbol{0}$。在这种情况下，Riccati 微分方程将简化成

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0} \tag{3-4-32}$$

该方程经常称作 Riccati 代数方程，相应的控制问题称为线性二次型最优调节问题 (LQ regulators，LQR)。假设 $\boldsymbol{u}^*(t) = -\boldsymbol{K}\boldsymbol{x}(t)$，其中 $\boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}$，则可以得出在状态反馈下的闭环系统的状态方程为 $(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}, \boldsymbol{B}, \boldsymbol{C} - \boldsymbol{D}\boldsymbol{K}, \boldsymbol{D})$。

Riccati 代数方程的求解可以由下面的语句直接求取

```
P=are(A,B*inv(R)*B',Q);  K=inv(R)*B'*P
```

状态反馈矩阵 $\boldsymbol{K}$ 和代数 Riccati 方程的解矩阵 $\boldsymbol{P}$ 也可以由控制系统工具箱的 lqr() 函数直接求出：[K,P]=lqr(A,B,Q,R)。

例 3-74 假设连续系统的状态方程模型参数为

$$A=\begin{bmatrix}2&0&4&1&2\\1&-2&-4&0&1\\1&4&3&0&2\\2&-2&2&3&3\\1&4&6&2&1\end{bmatrix},\quad B=\begin{bmatrix}1&2\\0&1\\0&0\\0&0\\0&0\end{bmatrix}$$

选择加权矩阵 $Q=\mathrm{diag}(1000,0,1000,500,500)$，$R=I_2$，则可以通过下面的语句直接设计出系统的状态反馈矩阵和 Riccati 方程的解为

```
>> A=[2,0,4,1,2; 1,-2,-4,0,1; 1,4,3,0,2; 2,-2,2,3,3; 1,4,6,2,1];
   B=[1,2; 0,1; 0,0; 0,0; 0,0]; % 状态方程
   Q=diag([1000 0 1000 500 500]); R=eye(2); % 加权矩阵输入
   [K,P]=lqr(A,B,Q,R)   % 状态反馈矩阵和 Riccati 方程的解
```

这样可以直接得出状态反馈矩阵 K 与 Riccati 方程的解矩阵为

$$K^{\mathrm{T}}=\begin{bmatrix}21.978&24.09\\-19.867&27.463\\-17.195&82.937\\15.978&75.931\\-7.1739&67.526\end{bmatrix},\quad P=\begin{bmatrix}21.978&-19.867&-17.195&15.978&-7.1739\\-19.867&67.198&117.33&43.975&81.874\\-17.195&117.33&503.52&345.84&237.17\\15.978&43.975&345.84&661.53&379.92\\-7.1739&81.874&237.17&379.92&374\end{bmatrix}$$

在该状态反馈下，可以由 eig(A-B*K) 语句直接得出闭环系统的极点为 $-70.9010, -5.9113, -2.1770, -5.8155\pm \mathrm{j}6.2961$。

5. 离散系统的二次型最优控制

对离散系统来说，二次型性能指标可以写成

$$J=\frac{1}{2}\sum_{k=0}^{N}\Big[x^{\mathrm{T}}(k)Qx(k)+u^{\mathrm{T}}(k)Ru(k)\Big] \tag{3-4-33}$$

其相应的动态 Riccati 方程为[22]

$$S(k)=F^{\mathrm{T}}\Big[S(k+1)-S(k+1)GR^{-1}G^{\mathrm{T}}S(k+1)\Big]F+Q \tag{3-4-34}$$

其中 $S(N)=Q$，N 为终止时刻，且 (F,G) 为离散状态方程矩阵。对二次型最优调节问题来说，S 为常数矩阵，这样离散 Riccati 代数方程为

$$S=F^{\mathrm{T}}\Big[S-SGR^{-1}G^{\mathrm{T}}S\Big]F+Q \tag{3-4-35}$$

这时控制律为

$$K=\Big[R+G^{\mathrm{T}}SG\Big]^{-1}B^{\mathrm{T}}SF \tag{3-4-36}$$

离散系统的代数 Riccati 方程可以由 dare() 函数求解，控制律 K 可以由 dlqr() 函数求解，其调用格式为 [K,S]=dlqr(F,G,Q,R)。

3.5 非线性运算与矩阵函数求值

3.5.1 面向矩阵元素的非线性运算

MATLAB 提供了大量函数，允许用户对矩阵进行处理，前面介绍的主要是矩阵的线性变换，本节将介绍如何对矩阵进行非线性运算。

事实上，MATLAB 提供了两类函数，其中一类是对矩阵的各个元素进行单独运算的，而另一类是对整个矩阵进行运算的。表 3-1 中列出各种对矩阵的各个元素单独进行非线性运算的函数，它们的调用方法是很显然的，其标准调用格式为 **B=函数名 (A)**，例如 $\boldsymbol{B}=\sin(\boldsymbol{A})$。

表 3-1 面向矩阵元素的非线性函数表

函数名	意 义	函数名	意 义
`abs()`	求模（绝对值）函数	`asin(), acos(), atan()`	反正弦、余弦、正切函数
`sqrt()`	求平方根函数	`log(), log10()`	自然和常用对数
`exp()`	指数函数	`real(), imag(), conj()`	求实虚部及共轭复数
`sin(),cos(),tan()`	正弦、余弦、正切函数	`round(),floor(),ceil()`	取整数函数

3.5.2 一般矩阵函数求值

面向矩阵的非线性运算有着广泛的应用前景，例如求取某个矩阵的指数可以直接应用于将连续状态方程模型变换成离散状态方程模型的转换过程。本节将介绍矩阵指数函数的运算方法，并以此为基础介绍矩阵三角函数求解的方法。本节还将给出通过 Jordan 变换求解一般矩阵函数的算法及其 MATLAB 实现。

1. 矩阵指数的运算

除了对矩阵的单个元素进行单独计算以外，一般还常常要求对整个矩阵做这样的非线性运算。例如，想求出一个矩阵的 e 指数，就需要用特殊的算法来完成。文献 [23] 中叙述了求解矩阵指数的 19 种不同方法，每一种方法都有自己的特点及适用范围。在 MATLAB 中提供的求取矩阵指数的函数 `expm()`，其调用格式为 **E=expm(A)**，该函数采用 Padé 近似技术来求取矩阵的指数。该函数还可以直接用于符号矩阵的求解。

例 3-75 考虑下面给出的矩阵 $\boldsymbol{A}=\begin{bmatrix}-2 & 1 & 0 & & \\ 0 & -2 & 1 & & \\ 0 & 0 & -2 & & \\ & & & -5 & 1 \\ & & & 0 & -5\end{bmatrix}$，试求出该矩阵的指数和对数，即 $\mathrm{e}^{\boldsymbol{A}}$ 和 $\ln \boldsymbol{A}$。

求解 如果对此矩阵进行指数运算和对数运算，则可以获得以下的结果:

```
>> Z=zeros(3,2); A=[[-2 1 0; 0 -2 1; 0 0 -2],Z; Z',[-5 1; 0 -5]];
   expm(A)    % 数值解求解
```

可以得出指数矩阵为

$$e^{A}=\begin{bmatrix}0.13533528 & 0.13533528 & 0.067667642 & 0 & 0\\ 0 & 0.13533528 & 0.13533528 & 0 & 0\\ 0 & 0 & 0.13533528 & 0 & 0\\ 0 & 0 & 0 & 0.006737947 & 0.006737947\\ 0 & 0 & 0 & 0 & 0.006737947\end{bmatrix}$$

原始问题还可以调用解析解函数 expm()，直接求解 e^{At}。注意，这里包含了变量 t，所以这是数值算法无法解出的。

```
>> syms t; expm(A*t)
```

这样可以得出状态转移矩阵为 $e^{At}=\begin{bmatrix}e^{-2t} & te^{-2t} & t^2e^{-2t}/2 & 0 & 0\\ 0 & e^{-2t} & te^{-2t} & 0 & 0\\ 0 & 0 & e^{-2t} & 0 & 0\\ 0 & 0 & 0 & e^{-5t} & te^{-5t}\\ 0 & 0 & 0 & 0 & e^{-5t}\end{bmatrix}$。

例 3-76 已知矩阵 $A\begin{bmatrix}-3 & -1 & -1\\ 0 & -3 & -1\\ 1 & 2 & 0\end{bmatrix}$，试求出 e^{At}。

求解 如果 Jordan 标准型不那么明显，则不能采用直接写出的方法求解 e^{At}，而应该采用广义特征向量矩阵的方式进行变换。现在考虑

```
>> syms t; A=[-3,-1,-1; 0,-3,-1; 1,2,0]; simple(expm(A*t))
```

这样可以得出该矩阵的状态转移矩阵

$$e^{At}=\begin{bmatrix}-e^{-2t}(-1+t) & -te^{-2t} & -te^{-2t}\\ -t^2e^{-2t}/2 & -e^{-2t}(-2+2t+t^2)/2 & -te^{-2t}(2+t)/2\\ te^{-2t}(2+t)/2 & te^{-2t}(t+4)/2 & e^{-2t}(2+t^2+4t)/2\end{bmatrix}$$

下面演示基于 Jordan 矩阵变换的 e^{At} 矩阵处理方法。

```
>> [V,J]=jordan(A)    % Jordan 矩阵变换
```

则可以得出 V 和 J 矩阵，并根据 J 写出 e^{Jt}

$$V=\begin{bmatrix}0 & -1 & 1\\ -1 & 0 & 0\\ 1 & 1 & 0\end{bmatrix},\ J=\begin{bmatrix}-2 & 1 & 0\\ 0 & -2 & 1\\ 0 & 0 & -2\end{bmatrix},\ e^{Jt}=\begin{bmatrix}e^{-2t} & te^{-2t} & t^2e^{-2t}/2\\ 0 & e^{-2t} & te^{-2t}\\ 0 & 0 & e^{-2t}\end{bmatrix}$$

这样，原矩阵的指数矩阵可以由指令 inv(V)*e^{Jt}*V 求出，其结果与直接求解的结果是完全一致的。

其实，用这样的方法求解矩阵指数不是此例子的目的，因为用符号运算工具箱中的 expm() 函数可以立即得出所需的结果。后面将通过例子演示其他函数，如正弦等函数如何用 Jordan 矩阵的方法求解。

2. 矩阵的三角函数的数值运算

MATLAB 下没有对矩阵进行三角函数运算的现成函数，求解其数值解可以通过 funm() 函数。该函数的目的是求出矩阵的任意函数，其调用方法为 $\boldsymbol{A}_1$=funm($\boldsymbol{A}$,'函数名')，其中，函数名应该由单引号括起来。例如，若想求出矩阵 $\boldsymbol{A}$ 的正弦矩阵，则可以使用命令 $\boldsymbol{B}$=funm($\boldsymbol{A}$,'sin')。

事实上，矩阵的非线性函数运算可以通过幂级数的方法简单地求出。例如，正弦函数可以由下面的幂级数展开式求出。

$$\sin \boldsymbol{A}=\sum_{i=0}^{\infty}(-1)^i\frac{\boldsymbol{A}^{2i+1}}{(2i+1)!}=\boldsymbol{A}-\frac{1}{3!}\boldsymbol{A}^3+\frac{1}{5!}\boldsymbol{A}^5+\cdots \tag{3-5-1}$$

可以用 MATLAB 实现正弦函数幂级数的展开。

```
function E=sinm1(A)
E=zeros(size(A)); F=A; k=1;
while norm(E+F-E,1)>0, E=E+F; F=-A^2*F/((k+2)*(k+1)); k=k+2; end
```

例 3-77　重新考虑例 3-75 中给出的矩阵，如果想对其中的 $\boldsymbol{A}$ 矩阵进行正弦运算，由上面的程序可以看出，看起来比较复杂的矩阵正弦函数的幂级数展开运算可以由几条 MATLAB 语句容易地编写出来。利用 $\boldsymbol{E}$=sinm1($\boldsymbol{A}$) 函数可以容易地求出原矩阵 $\boldsymbol{A}$ 的正弦矩阵为

$$\sin \boldsymbol{A}=\begin{bmatrix}-0.90929743 & -0.41614684 & 0.45464871 & 0 & 0\\ 0 & -0.90929743 & -0.41614684 & 0 & 0\\ 0 & 0 & -0.90929743 & 0 & 0\\ 0 & 0 & 0 & 0.95892427 & 0.28366219\\ 0 & 0 & 0 & 0 & 0.95892427\end{bmatrix}$$

可以测出，该函数一共进行了 39 次叠加运算。

3. 矩阵三角函数的解析运算

再考虑矩阵三角函数的解析解求解方法。先考虑标量三角函数的运算公式，根据著名的 Euler 公式 $\mathrm{e}^{\mathrm{j}a}=\cos a+\mathrm{j}\sin a$ 与 $\mathrm{e}^{-\mathrm{j}a}=\cos a-\mathrm{j}\sin a$ 可以立即导出

$$\sin a=\frac{1}{\mathrm{j}2}(\mathrm{e}^{\mathrm{j}a}-\mathrm{e}^{-\mathrm{j}a}),\quad \cos a=\frac{1}{2}(\mathrm{e}^{\mathrm{j}a}+\mathrm{e}^{-\mathrm{j}a}) \tag{3-5-2}$$

此公式可以直接用于 a 为矩阵的形式。下面通过例子演示一般矩阵的正弦和余弦函数的解析解运算。

例 3-78 仍考虑例 3-75 中给出的矩阵，试求解 $\sin \boldsymbol{A}$。

求解 可以利用现成的 expm() 函数求出矩阵的正弦函数。

```
>> Z=zeros(3,2); A=[[-2 1 0; 0 -2 1; 0 0 -2],Z; Z',[-5 1; 0 -5]];
   j=sqrt(-1); A1=(expm(A*j)-expm(-A*j))/(2*j),
```

得出的解与例 3-77 完全一致，证明该解是正确的。

例 3-79 假设给出如下的矩阵 $\boldsymbol{A}=\begin{bmatrix}-7 & 2 & 0 & -1\\ 1 & -4 & 2 & 1\\ 2 & -1 & -6 & -1\\ -1 & -1 & 0 & -4\end{bmatrix}$，已知该矩阵有重特征值，试求出该矩阵的正弦函数 $\sin \boldsymbol{A}t$ 和余弦函数 $\cos \boldsymbol{A}t$。

求解 根据式 (3-5-2) 可以由下面的语句求解矩阵的正弦和余弦函数

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];
   syms t; j=sym(sqrt(-1));
   A1=simple((expm(A*j*t)-expm(-A*j*t))/(2*j)),
   A2=simple((expm(A*j*t)+expm(-A*j*t))/2)
```

由于结果过于冗长，这里只列出其正弦函数的结果如下：

$$\sin \boldsymbol{A}t=\left[\begin{array}{ll}-2/9\sin 3t+(t^2-7/9)\sin 6t-5/3t\cos 6t & -\sin 3t/3+\sin 6t/3+t\cos 6t\\ -2/9\sin 3t+(t^2+2/9)\sin 6t+t\cos 6t/3 & -\sin 3t/3-2/3\sin 6t+t\cos 6t\\ -2/9\sin 3t+(-2t^2+2/9)\sin 6t+4/3t\cos 6t & -\sin 3t/3+\sin 6t/3-2t\cos 6t\\ 4/9\sin 3t+(t^2-4/9)\sin 6t+t\cos 6t/3 & 2/3\sin 3t-2/3\sin 6t+t\cos 6t\end{array}\right.$$

$$\left.\begin{array}{ll}-2/9\sin 3t+(2/9+t^2)\sin 6t-2/3t\cos 6t & \sin 3t/9+(-1/9+t^2)\sin 6t-2/3t\cos 6t\\ -2/9\sin 3t+(2/9+t^2)\sin 6t+4/3t\cos 6t & \sin 3t/9+(-1/9+t^2)\sin 6t+4/3t\cos 6t\\ -2/9\sin 3t-(7/9+2t^2)\sin 6t-2/3t\cos 6t & \sin 3t/9-(1/9+2t^2)\sin 6t-2/3t\cos 6t\\ 4/9\sin 3t+(-4/9+t^2)\sin 6t+4/3t\cos 6t & -2/9\sin 3t+(-7/9+t^2)\sin 6t+4/3t\cos 6t\end{array}\right]$$

4. 一般矩阵函数的运算

除了对整个矩阵求取矩阵指数之外，MATLAB 还允许求取矩阵的其他非线性函数，其中常用的函数还有 logm() (矩阵求对数)、sqrtm() (矩阵求平方根) 和 funm() (矩阵求任意函数) 等。可以看出，这里的函数名很有特点，每个函数名在标准函数名的后面加了一个后缀 m，表示对矩阵而不是对矩阵元素进行运算。

遗憾的是，现有的 funm() 函数是基于特征值和特征向量矩阵的，所以矩阵有重根时，由于特征向量矩阵奇异，故得出的结果是不可靠的，甚至是错误的。这里将介绍基于 Jordan 矩阵的矩阵函数求解方法[2]。

首先可以将 m_i 阶 Jordan 块 $\boldsymbol{J}_i$ 写成 $\boldsymbol{J}_i=\lambda_i\boldsymbol{I}+\boldsymbol{H}_{m_i}$，其中，$\lambda_i$ 为 Jordan 矩阵的重特征值，$\boldsymbol{H}_{m_i}$ 为幂零矩阵，即 $k\geqslant m_i$ 时 $\boldsymbol{H}_{m_i}^k\equiv\boldsymbol{0}$。这样可以证明，矩阵函数 $\psi(\boldsymbol{J}_i)$ 可以由下式求出。

$$\psi(\boldsymbol{J}_i)=\psi(\lambda_i)\boldsymbol{I}_{m_i}+\psi'(\lambda_i)\boldsymbol{H}_{m_i}+\cdots+\frac{\psi^{(m_i-1)}(\lambda_i)}{(m_i-1)!}\boldsymbol{H}_{m_i}^{m_i-1} \qquad (3\text{-}5\text{-}3)$$

如果通过 Jordan 矩阵分解的方法可以将任意矩阵 $\boldsymbol{A}$ 分解成

$$\boldsymbol{A}=\boldsymbol{V}\begin{bmatrix}\boldsymbol{J}_1 & & & \\ & \boldsymbol{J}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{J}_m\end{bmatrix}\boldsymbol{V}^{-1} \tag{3-5-4}$$

这样，该矩阵的任意函数 $\psi(\boldsymbol{A})$ 可以最终如下求出。如果通过 Jordan 矩阵分解的方法可以将任意矩阵 $\boldsymbol{A}$ 分解成

$$\psi(\boldsymbol{A})=\boldsymbol{V}\begin{bmatrix}\psi(\boldsymbol{J}_1) & & & \\ & \psi(\boldsymbol{J}_2) & & \\ & & \ddots & \\ & & & \psi(\boldsymbol{J}_m)\end{bmatrix}\boldsymbol{V}^{-1} \tag{3-5-5}$$

根据上面的算法可以立即编写出新的 `funm()` 函数，应该置于 `@sym` 目录下，可以推导任意矩阵函数的解析解。该函数的清单为 [4]

```
function F=funm(A,fun,x)
[V,J]=jordan(A); v1=[0,diag(J,1)']; v2=[find(v1==0),length(v1)+1];
for i=1:length(v2)-1
    v_lambda(i)=J(v2(i),v2(i)); v_n(i)=v2(i+1)-v2(i);
end
m=length(v_lambda); F=sym([]);
for i=1:m
    J1=J(v2(i):v2(i)+v_n(i)-1,v2(i):v2(i)+v_n(i)-1);
    fJ=funJ(J1,fun,x); F=diagm(F,fJ);
end
F=V*F*inv(V);
function fJ=funJ(J,fun,x)
lam=J(1,1); f1=fun; fJ=subs(fun,x,lam)*eye(size(J));
H=diag(diag(J,1),1); H1=H;
for i=2:length(J)
   f1=diff(f1,x); a1=subs(f1,x,lam); fJ=fJ+a1*H1; H1=H1*H/i;
end
```

该函数的调用格式为 $\boldsymbol{A}_1$=`funm(`$\boldsymbol{A}$`,funx,`x`)`，其中，x 为符号型自变量，`funx` 为 x 的函数表示。例如，若想求出 $\mathrm{e}^{\boldsymbol{A}}$，则可以将 `funx` 填写成 `exp(x)`。其实，`funx` 参数可以描述任意复杂的函数，如 exp(x*t) 表示求取 $\mathrm{e}^{\boldsymbol{A}t}$，其中 t 也应该事先设置成符号变量。另外，该函数还可以表示成 `exp(x*cos(x*t))` 型的复合函数，表示需要求取 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$。

例 3-80 已知矩阵 $\boldsymbol{A}=\begin{bmatrix}-7 & 2 & 0 & -1\\ 1 & -4 & 2 & 1\\ 2 & -1 & -6 & -1\\ -1 & -1 & 0 & -4\end{bmatrix}$，试求出矩阵函数 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$。

求解 可以用下面的语句将其输入到 MATLAB 环境中。

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];
```

如果想求出 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$，分析题意，可以用下面的语句

```
>> syms x t; A1=funm(sym(A),exp(x*cos(x*t)),x)
```

得出的结果是很冗长的，这里只给出其中一项为

$$\begin{aligned}\psi_{1,1}(\boldsymbol{A})=&\,2/9\mathrm{e}^{-3\cos 3t}+(2t\sin 6t+6t^2\cos 6t)\mathrm{e}^{-6\cos 6t}+(\cos 6t-6t\sin 6t)^2\mathrm{e}^{-6\cos 6t}\\&-5/3(\cos 6t-6t\sin 6t)\mathrm{e}^{-6\cos 6t}+7/9\mathrm{e}^{-6\cos 6t}\end{aligned}$$

可见，这样得出的 $\psi_{1,1}(t)$ 有很多项均是 $\mathrm{e}^{-6\cos 6t}$ 的系数项，故可以通过合并同类项的化简方法手动给出下面的命令：

```
>> collect(A1(1,1),exp(-6*cos(6*t)))
```

则可以得出如下的化简结果：

$$\psi_{1,1}(\boldsymbol{A})=\left[12t\sin 6t+6t^2\cos 6t+(\cos 6t-6t\sin 6t)^2-\frac{5}{3}\cos 6t+\frac{7}{9}\right]\mathrm{e}^{-6\cos 6t}+\frac{2}{9}\mathrm{e}^{-3\cos 3t}$$

进一步地，若令 $t=1$，则 subs($\boldsymbol{A}_1$,t,1) 可以求出 $\mathrm{e}^{\boldsymbol{A}\cos\boldsymbol{A}}$ 的精确数值解为

$$\mathrm{e}^{\boldsymbol{A}\cos\boldsymbol{A}}=\begin{bmatrix}4.3583154 & 6.504410916 & 4.363467442 & -2.132643538\\ 4.371767378 & 6.507558809 & 4.380067313 & -2.116043667\\ 4.265287076 & 6.479511109 & 4.251835099 & -2.247423775\\ -8.620454583 & -12.98392202 & -8.612154647 & 4.383215207\end{bmatrix}$$

3.5.3 矩阵函数求值在控制系统中的应用

前面指出了矩阵指数函数在线性系统离散化中的应用，这里将给出具体的应用实例，并介绍离散状态方程模型的连续化方法。以矩阵指数的解析运算为基础还可以得出系统的状态转移矩阵，从而得出线性系统时域响应的解析解。这里将通过例子介绍求解的全过程。

1. 连续系统的离散化

假设连续系统的状态方程模型由式 (3-1-8) 给出，则可以写出状态变量的解析解为

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{3-5-6}$$

选择采样周期为 T，对之进行离散化，可以选择 $t_0=kT$, $t=(k+1)T$，可得

$$\boldsymbol{x}[(k+1)T]=\mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT)+\int_{kT}^{(k+1)T}\mathrm{e}^{\boldsymbol{A}[(k+1)T-\tau]}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{3-5-7}$$

考虑对输入信号采用零阶保持器，亦即在同一采样周期内输入信号的值保持不变。假设在采样周期内输入信号为固定的值 $\boldsymbol{u}(kT)$，故上式可以化简为

$$\boldsymbol{x}[(k+1)T] = \mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_0^T \mathrm{e}^{\boldsymbol{A}\tau}\mathrm{d}\tau\, \boldsymbol{B}\boldsymbol{u}(kT) \tag{3-5-8}$$

对照式 (3-5-8) 与式 (3-1-9)，可以发现，使用零阶保持器后连续系统离散化可以直接获得离散状态方程模型，离散后系统的参数可以由下式求出

$$\boldsymbol{F} = \mathrm{c}^{\boldsymbol{A}T},\ \ \boldsymbol{G} = \int_0^T \mathrm{e}^{\boldsymbol{A}\tau}\mathrm{d}\tau\, \boldsymbol{B} \tag{3-5-9}$$

且二者的 $\boldsymbol{C}$ 与 $\boldsymbol{D}$ 矩阵完全一致。当然，$\boldsymbol{F}$ 矩阵可以通过 `expm()` 函数直接求出，而 $\boldsymbol{G}$ 矩阵的求取较麻烦。MATLAB 的控制系统工具箱提供了很实用的 `c2d()` 函数，可以由 G_{d}=`c2d(`G,T`)`，其中 G 为连续系统模型，T 为采样周期，而得出的 G_{d} 为变换出的离散系统模型。

例 3-81 考虑例 3-8 中给出的多变量状态方程模型，假设采样周期 $T=0.1$ 秒，则可以用下面的命令将模型输入到 MATLAB 工作空间，并得出离散化的状态方程模型。

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
      6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5]; C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D); T=0.1; Gd=c2d(G,T)
```

对连续状态方程模型进行离散化，则得出离散系统数学表示为

$$\begin{cases} \boldsymbol{x}_{k+1} = \begin{bmatrix} -0.15 & -1.6481 & -1.6076 & -1.14 \\ 0.5735 & 1.822 & 0.8018 & 0.5735 \\ 0.5765 & 0.8362 & 1.8059 & 0.5765 \\ -0.5665 & -0.8261 & -0.7959 & 0.4236 \end{bmatrix}\boldsymbol{x}_k + \begin{bmatrix} -0.1842 & -0.1272 \\ 0.2668 & 0.1036 \\ 0.3679 & 0.174 \\ -0.1657 & -0.02326 \end{bmatrix}\boldsymbol{u}_k \\ \boldsymbol{y}_k = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix}\boldsymbol{x}_k \end{cases}$$

2. 离散状态方程的连续化

若给出离散状态方程模型 (3-5-7)，则在采样周期 T 下获得相应连续状态方程模型 (3-5-6) 的过程为离散系统的连续化。由变换式 (3-5-8) 可见，其反变换可以直接由下式得出 [24]

$$\boldsymbol{A} = \frac{1}{T}\ln\boldsymbol{F},\ \ \boldsymbol{B} = (\boldsymbol{F}-\boldsymbol{I})^{-1}\boldsymbol{A}\boldsymbol{G} \tag{3-5-10}$$

这里涉及到矩阵求对数的过程。矩阵的对数可以由 `logm()` 函数求出，也可以由前面介绍的 `funm()` 函数求出。在 MATLAB 环境中，可以利用其控制系统工具

箱中提供的 G_1=d2c(G) 函数进行连续化变换，其中在调用语句中无需再申明采样周期信息，因为该信息已经包含在零散模型 G 中。利用该语句即可以得出相应的连续系统模型 G_1，该函数同样适用于带有时间延迟系统。

例 3-82 考虑例 3-81 中给出的原问题。前面已经得出了离散化的状态方程模型，试用连续化方法得出其连续模型，观察是否可以恢复原来的连续模型。

求解 由下面的语句可以直接求取对应的连续模型，可见得出的结果与已知的连续模型是完全一致的。

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6;
        6,8.7,8.4,6; -5.9,-8.6,-8.3,-6];
   B=[1.5,0.2; 1,0.3; 2,1; 0,0.5]; C=[2,0.5,0,0.8; 0.3,0.3,0.2,1];
   D=zeros(2,2); G=ss(A,B,C,D); T=0.1; Gd=c2d(G,T); G2=d2c(Gd)
```

如果采用底层命令，则下面的语句也能得出完全一致的结果。

```
>> A1=logm(Gd.a)/T; B1=inv(Gd.a-eye(size(Gd.a)))*A1*Gd.b
```

3. 连续状态方程的解析解

在某给定信号 $\boldsymbol{u}(t)$ 的激励下，其状态变量的解析解可以由式 (3-5-6) 得出。对于一般的输入信号来说，直接由该式求取系统的解析解并非很容易的事，因为其中积分项不是很好处理。如果能对状态方程进行某种变换，消去输入信号，则该方程的解析解就容易求解了。这里将对一类典型输入信号介绍状态增广的方法，将其化为不含有输入信号的状态方程，从而直接求解原来状态方程的解析解[25]。

先考虑单位阶跃信号 $u(t)=1(t)$，若假设有另外一个状态变量 $x_{n+1}(t)=u(t)$，则其导数为 $\dot{x}_{n+1}(t)=0$，这样系统的状态方程可以改写为

$$\begin{bmatrix}\dot{\boldsymbol{x}}(t)\\ \hdashline \dot{x}_{n+1}(t)\end{bmatrix}=\left[\begin{array}{c:c}\boldsymbol{A} & \boldsymbol{B}\\ \hdashline \boldsymbol{0} & \boldsymbol{0}\end{array}\right]\begin{bmatrix}\boldsymbol{x}(t)\\ \hdashline x_{n+1}(t)\end{bmatrix} \tag{3-5-11}$$

可见，这样就把原始的状态方程转换成直接可以求解的自治系统方程了

$$\begin{cases}\dot{\widetilde{\boldsymbol{x}}}(t)=\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}}(t)\\ \widetilde{\boldsymbol{y}}(t)=\widetilde{\boldsymbol{C}}\widetilde{\boldsymbol{x}}(t)\end{cases} \tag{3-5-12}$$

式中 $\widetilde{\boldsymbol{x}}^{\mathrm{T}}(t)=[\boldsymbol{x}^{\mathrm{T}}(t),x_{n+1}(t)]$，且 $\widetilde{\boldsymbol{x}}^{\mathrm{T}}(0)=[\boldsymbol{x}^{\mathrm{T}}(0),1]$，其解析解比较容易求出

$$\widetilde{\boldsymbol{x}}(t)=\mathrm{e}^{\widetilde{\boldsymbol{A}}t}\widetilde{\boldsymbol{x}}(0) \tag{3-5-13}$$

对可以实现这样变换的输入信号进行扩展，则可以定义一类典型输入信号

$$u(t)=u_1(t)+u_2(t)=\sum_{i=0}^{m}c_it^i+\mathrm{e}^{d_1t}\Big[d_2\cos(d_4t)+d_3\sin(d_4t)\Big] \tag{3-5-14}$$

引入附加状态变量 $x_{n+1}=\mathrm{e}^{d_1t}\cos(d_4t)$, $x_{n+2}=\mathrm{e}^{d_1t}\sin(d_4t)$, $x_{n+3}=u_1(t),\cdots$, $x_{n+m+3}=u_1^{(m-1)}(t)$，通过推导，则可以得出式 (3-5-12) 中给出的系统增广状态方程模型，式中

$$\widetilde{\boldsymbol{A}}=\left[\begin{array}{c|cc|cccc}\boldsymbol{A} & d_2\boldsymbol{B} & d_3\boldsymbol{B} & \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \hline \boldsymbol{0} & d_1 & -d_4 & & \boldsymbol{0} & & \\ & d_4 & d_1 & & & & \\ \hline & & & 0 & 1 & \cdots & 0\\ \boldsymbol{0} & & \boldsymbol{0} & 0 & 0 & \cdots & 0\\ & & & \vdots & \vdots & \ddots & \vdots\\ & & & 0 & 0 & \cdots & 0\end{array}\right],\ \widetilde{\boldsymbol{x}}(t)=\left[\begin{array}{c}\boldsymbol{x}(t)\\ \hline x_{n+1}(t)\\ x_{n+2}(t)\\ \hline x_{n+3}(t)\\ x_{n+4}(t)\\ \vdots\\ x_{n+m+3}(t)\end{array}\right],\ \widetilde{\boldsymbol{x}}(0)=\left[\begin{array}{c}\boldsymbol{x}(0)\\ \hline 1\\ 0\\ \hline c_0\\ c_1\\ \vdots\\ c_m m!\end{array}\right] \tag{3-5-15}$$

这样系统的状态方程模型的解析解为

$$\widetilde{\boldsymbol{x}}(t)=\mathrm{e}^{\widetilde{\boldsymbol{A}}t}\widetilde{\boldsymbol{x}}(0) \tag{3-5-16}$$

作者用 MATLAB 语言编写了一个函数 ss_augment()，可以用来求取系统的增广状态方程模型，该函数的内容如下：

```
function [Ga,Xa]=ss_augment(G,cc,dd,X)
G=ss(G); Aa=G.a; Ca=G.c; Xa=X; Ba=G.b; D=G.d;
if (length(dd)>0 & sum(abs(dd))>1e-5),
   if (abs(dd(4))>1e-5),
      Aa=[Aa dd(2)*Ba, dd(3)*Ba; ...
          zeros(2,length(Aa)), [dd(1),-dd(4); dd(4),dd(1)]];
      Ca=[Ca dd(2)*D dd(3)*D]; Xa=[Xa; 1; 0]; Ba=[Ba; 0; 0];
   else,
      Aa=[Aa dd(2)*B; zeros(1,length(Aa)) dd(1)];
      Ca=[Ca dd(2)*D]; Xa=[Xa; 1]; Ba=[B;0];
   end
end
if (length(cc)>0 & sum(abs(cc))>1e-5), M=length(cc);
   Aa=[Aa Ba zeros(length(Aa),M-1); zeros(M-1,length(Aa)+1) ...
        eye(M-1); zeros(1,length(Aa)+M)];
   Ca=[Ca D zeros(1,M-1)]; Xa=[Xa; cc(1)]; ii=1;
   for i=2:M, ii=ii*i; Xa(length(Aa)+i)=cc(i)*ii;
end, end
Ga=ss(Aa,zeros(size(Ca')),Ca,D);
```

其中 cc= $[c_0,c_1,\cdots,c_m]$，且 dd=$[d_1,d_2,d_3,d_4]$。构造出系统的增广状态方程模型后，则可以用 MATLAB 符号运算工具箱的 expm() 函数求取各个状态变量的解析解。

例 3-83 系统的状态方程模型为

$$\begin{cases}\dot{\boldsymbol{x}}(t)=\begin{bmatrix}-19 & -16 & -16 & -19\\ 21 & 16 & 17 & 19\\ 20 & 17 & 16 & 20\\ -20 & -16 & -16 & -19\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}1\\0\\1\\2\end{bmatrix}u(t)\\ y(t)=[2,1,0,0]\,\boldsymbol{x}(t)\end{cases}$$

其中状态变量初值为 $\boldsymbol{x}^{\mathrm{T}}(0)=[0,1,1,2]$。假设系统的输入信号为 $u(t)=2+2\mathrm{e}^{-3t}\sin 2t$，试求出该方程的解析解。

求解 分析原问题，可以构造出多项式系数向量 cc 和指数向量 dd，这样可以由 ss_augment() 函数得出系统的增广状态方程模型

```
>> cc=[2]; dd=[-3,0,2,2]; x0=[0; 1; 1; 2];
   A=[-19,-16,-16,-19; 21,16,17,19; 20,17,16,20; -20,-16,-16,-19];
   B=[1; 0; 1; 2]; C=[2 1 0 0]; D=0; G=ss(A,B,C,D);
   [Ga,xx0]=ss_augment(G,cc,dd,x0); Ga.a, xx0'
```

由得出的结果可以直接写出系统的增广状态方程模型及初值为

$$\dot{\tilde{\boldsymbol{x}}}(t)=\left[\begin{array}{cccc|cc|c}-19 & -16 & -16 & -19 & 0 & 2 & 1\\ 21 & 16 & 17 & 19 & 0 & 0 & 0\\ 20 & 17 & 16 & 20 & 0 & 2 & 1\\ -20 & -16 & -16 & -19 & 0 & 4 & 2\\ \hline 0 & 0 & 0 & 0 & -3 & -2 & 0\\ 0 & 0 & 0 & 0 & 2 & -3 & 0\\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0\end{array}\right]\tilde{\boldsymbol{x}}(t),\quad \tilde{\boldsymbol{x}}(0)=\left[\begin{array}{c}0\\1\\1\\2\\ \hline 1\\0\\ \hline 2\end{array}\right]$$

得出了系统的增广状态方程模型，则可以用下面的语句直接获得生成信号的解析解

```
>> syms t; y=Ga.c*expm(Ga.a*t)*xx0; % 求解系统的解析解
```

其解析解的数学形式可以写成

$$y(t)=-54+\frac{127}{4}t\mathrm{e}^{-t}+57\mathrm{e}^{-3t}+\frac{119}{8}\mathrm{e}^{-t}+4t^2\mathrm{e}^{-t}-\frac{135}{8}\mathrm{e}^{-3t}\cos 2t+\frac{77}{4}\mathrm{e}^{-3t}\sin 2t$$

3.5.4 基于矩阵积分的线性微分方程求解

若已知系统的状态方程模型 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$，则其解析解可以表示成

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{3-5-17}$$

3.5.3 节介绍了用状态增广方法求系统解析解的方法。同样的问题还可以利用矩阵积分的直接方法来求解，得出原系统的解析解。

例 3-84 考虑例 3-83 中的系统模型，试利用直接积分的方法求其解析解。

求解 直接利用式 (3-5-17)，可以由下面语句直接求解系统的解析解

```
>> syms t tau; u=2+2*exp(-3*tau)*sin(2*tau);
   A=[-19,-16,-16,-19; 21,16,17,19; 20,17,16,20; -20,-16,-16,-19];
   B=[1; 0; 1; 2]; C=[2 1 0 0]; x0=[0; 1; 1; 2];
   y=C*(expm(A*t)*x0+int(expm(A*(t-tau))*B*u,tau,0,t)); simple(y)
```

可以得出系统响应的解析解为

$$y(t)=\frac{1}{8}\left(591+119\mathrm{e}^{2t}+254t\mathrm{e}^{2t}+308\sin t\cos t-270\cos^2 t-432\mathrm{e}^{3t}+32t^2\mathrm{e}^{2t}\right)\mathrm{e}^{-3t}$$

得出的解在表示形式上和例 3-83 不同，但可以验证，二者的结果完全相同，例如，将 e^{-3t} 遍乘括号中各项，则

$$y(t)=\frac{1}{8}\left(591\mathrm{e}^{-3t}+119\mathrm{e}^{-t}+254t\mathrm{e}^{-t}+154\mathrm{e}^{-3t}\sin 2t-270\mathrm{e}^{-3t}\cos^2 t-432+32t^2\mathrm{e}^{-t}\right)$$

3.6 习题与思考题

1 Jordan 矩阵是矩阵分析中一类很实用的矩阵，其一般形式为

$$\boldsymbol{J}=\begin{bmatrix}-\alpha & 1 & 0 & \cdots & 0\\ 0 & -\alpha & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & -\alpha\end{bmatrix},\quad 例如\ \boldsymbol{J}_1=\begin{bmatrix}-5 & 1 & 0 & 0 & 0\\ 0 & -5 & 1 & 0 & 0\\ 0 & 0 & -5 & 1 & 0\\ 0 & 0 & 0 & -5 & 1\\ 0 & 0 & 0 & 0 & -5\end{bmatrix}$$

试利用 diag() 函数给出构造 $\boldsymbol{J}_1$ 的语句。

2 幂零矩阵是一类特殊的矩阵，其基本形式为

$$\boldsymbol{H}_n=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 0 & 0 & 0 & \cdots & 0\end{bmatrix}$$

亦即，矩阵的次主对角线元素为 1，其余均为 0，试验证对指定阶次的幂零矩阵，有 $\boldsymbol{H}_n^i=\boldsymbol{0}$ 对所有的 $i\geqslant n$ 成立。

3 试从矩阵的显示格式区分符号矩阵和数值矩阵，明确它们的含义和应用场合。若 $\boldsymbol{A}$ 矩阵为数值矩阵，$\boldsymbol{B}$ 为符号矩阵，$\boldsymbol{C}$=$\boldsymbol{A}$*$\boldsymbol{B}$ 运算得出的 $\boldsymbol{C}$ 矩阵是符号矩阵还是数值矩阵？

4 请将矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 输入到 MATLAB 环境中，并将它们转换成符号矩阵。

$$\boldsymbol{A}=\begin{bmatrix}5&7&6&5&1&6&5\\2&3&1&0&0&1&4\\6&4&2&0&6&4&4\\3&9&6&3&6&6&2\\10&7&6&0&0&7&7\\7&2&4&4&0&7&7\\4&8&6&7&2&1&7\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}3&5&5&0&1&2&3\\3&2&5&4&6&2&5\\1&2&1&1&3&4&6\\3&5&1&5&2&1&2\\4&1&0&1&2&0&1\\-3&-4&-7&3&7&8&12\\1&-10&7&-6&8&1&5\end{bmatrix}$$

5 试求出 Vandermonde 矩阵 $\boldsymbol{A}=\begin{bmatrix}a^4&a^3&a^2&a&1\\b^4&b^3&b^2&b&1\\c^4&c^3&c^2&c&1\\d^4&d^3&d^2&d&1\\e^4&e^3&e^2&e&1\end{bmatrix}$ 的行列式，并以最简的形式显示结果。

6 假设线性系统由下面的常微分方程给出

$$\begin{cases}\dot{x}_1(t)=-x_1(t)+x_2(t)\\ \dot{x}_2(t)=-x_2(t)-3x_3(t)+u_1(t)\\ \dot{x}_3(t)=-x_1(t)-5x_2(t)-3x_3(t)+u_2(t)\\ y=-x_2(t)+u_1(t)-5u_2(t)\end{cases}$$

式中有两个输入信号 $u_1(t)$ 与 $u_2(t)$，请在 MATLAB 工作空间中表示这个双输入系统模型，并由得出的状态方程模型求出等效的传递函数模型，并观察其传递函数的形式。

7 利用 MATLAB 语言提供的现成函数对习题 4 中给出的两个矩阵进行分析，判定它们是否为奇异矩阵，得出矩阵的秩、行列式、迹和逆矩阵，检验得出的逆矩阵是否正确。

8 试求出习题 4 中给出的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的特征多项式、特征值与特征向量，并验证 Hamilton-Caylay 定理，解释并验证如何运算能消除误差。

9 试对习题 4 中给出的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵进行奇异值分解、LU 分解及正交分解矩阵。

10 试求出下面矩阵的特征值、特征向量、奇异值。

$$\boldsymbol{A}=\begin{bmatrix}2&7&5&7&7\\7&4&9&3&3\\3&9&8&3&8\\5&9&6&3&6\\2&6&8&5&4\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}703&795&980&137&661\\547&957&271&12&284\\445&523&252&894&469\\695&880&876&199&65\\621&173&737&299&988\end{bmatrix}$$

11 给出线性系统的状态方程模型，请判定系统的稳定性。

$$①\ \dot{\boldsymbol{x}}(t)=\begin{bmatrix}-0.2 & 0.5 & 0 & 0 & 0\\ 0 & -0.5 & 1.6 & 0 & 0\\ 0 & 0 & -14.3 & 85.8 & 0\\ 0 & 0 & 0 & -33.3 & 100\\ 0 & 0 & 0 & 0 & -10\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\0\\0\\0\\30\end{bmatrix}u(t)$$

$$②\ \boldsymbol{x}[(k+1)T]=\begin{bmatrix}17 & 24.54 & 1 & 8 & 15\\ 23.54 & 5 & 7 & 14 & 16\\ 4 & 6 & 13.75 & 20 & 22.5889\\ 10.8689 & 1.2900 & 19.099 & 21.896 & 3\\ 11 & 18.0898 & 25 & 2.356 & 9\end{bmatrix}\boldsymbol{x}(kT)+\begin{bmatrix}1\\2\\3\\4\\5\end{bmatrix}u(kT)$$

12 判定下列系统的可控、可观测性，求出它们的可控、可观测及 Luenberger 标准型实现，并求出系统的 2-范数和无穷范数。

$$①\ \boldsymbol{A}=\begin{bmatrix}0 & 1 & 1 & 1\\ 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 1\\ 1 & 0\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$$

$$②\ \boldsymbol{A}=\begin{bmatrix}0 & 2 & 0 & 0\\ 0 & 1 & -2 & 0\\ 0 & 0 & 3 & 1\\ 1 & 0 & 0 & 0\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}2 & 0\\ 1 & 2\\ 0 & 1\\ 0 & 0\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$$

13 求出下面状态方程模型的最小实现。

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0 & -3 & 0 & 0\\ 1 & -4 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -2\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}3 & 2\\ 1 & 2\\ 1 & 1\\ 1 & 1\end{bmatrix}\boldsymbol{u}(t),\ \ \boldsymbol{y}(t)=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\boldsymbol{x}(t)$$

14 假设系统的状态方程模型为

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}-0.2 & 0.5 & 0 & 0 & 0\\ 0 & -0.5 & 1.6 & 0 & 0\\ 0 & 0 & -14.3 & 85.8 & 0\\ 0 & 0 & 0 & -33.3 & 100\\ 0 & 0 & 0 & 0 & -10\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\0\\0\\0\\30\end{bmatrix}u(t),\ y(t)=[1,0,0,0,0]\boldsymbol{x}(t)$$

请求出系统所有的零点和极点。如果想将其极点配置到 $\boldsymbol{P}=[-1,-2,-3,-4,-5]$，请按状态反馈的方式设计出控制器实现闭环极点的移动。如果想再进一步改进闭环系统的动态响应，则可以修正期望闭环极点的位置，然后进行重新设计。

15 对给定的对象模型

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}2 & 1 & 0 & 0\\ 0 & 2 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\1\\1\\1\end{bmatrix}u(t),\ y(t)=[1,0,1,0]\boldsymbol{x}(t)$$

请设计出一个状态反馈向量 $\boldsymbol{k}$，使得闭环系统的极点配置到 $(-2,-2,-1,-1)$ 位置。另外，如果想将系统的所有极点均配置到 -2，这样的配置是否可行？请解释原因。

16 对下列各个开环模型进行频域分析，绘制出 Bode 图、Nyquist 图及 Nichols 图，并求出系统的幅值裕度和相位裕度，在各个图形上标注出来。假设闭环系统由单位负反馈构造而成，试由频域分析判定闭环系统的稳定性，并用阶跃响应来验证。

① $G(s)=\dfrac{8(s+1)}{s^2(s+15)(s^2+6s+10)}$，② $G(s)=\dfrac{4(s/3+1)}{s(0.02s+1)(0.05s+1)(0.1s+1)}$

③ $\begin{cases}\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0&2&1\\-3&-2&0\\1&3&4\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}4\\3\\2\end{bmatrix}u(t)\\ y(t)=[1,2,3]\boldsymbol{x}(t)\end{cases}$

④ $H(z)=0.45\dfrac{(z+1.31)(z+0.054)(z-0.957)}{z(z-1)(z-0.368)(z-0.99)}$，

⑤ $G(s)=\dfrac{6(-s+4)}{s^2(0.5s+1)(0.1s+1)}$，⑥ $G(s)=\dfrac{10s^3-60s^2+110s+60}{s^4+17s^3+82s^2+130s+100}$

17 试求下面线性代数方程的解析解与数值解，并检验解的正确性。

$$\begin{bmatrix}2&-9&3&-2&-1\\10&-1&10&5&0\\8&-2&-4&-6&3\\-5&-6&-6&-8&-4\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}-1&-4&0\\-3&-8&-4\\0&3&3\\9&-5&3\end{bmatrix}$$

18 假设带有时间延迟的系统传递函数矩阵为

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.06371}{s^2+2.517s+0.5618}\mathrm{e}^{-0.72s} & \dfrac{0.4464}{s+0.4831}\\ \dfrac{0.9357}{s^2+3.019s+2.77}\mathrm{e}^{-0.3s} & \dfrac{-0.1085}{s+0.3413}\mathrm{e}^{-1.29s}\end{bmatrix}$$

试绘制其带有 Gershgorin 带的逆 Nyquist 阵列，分析其是否为对角占优的系统，绘制系统的开环阶跃响应，该响应是否符合你的结论？

19 考虑下面给出的双输入双输出系统

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.806s+0.264}{s^2+1.15s+0.202} & \dfrac{-(15s+1.42)}{s^3+12.8s^2+13.6s+2.36}\\ \dfrac{1.95s^2+2.12s+4.90}{s^3+9.15s^2+9.39s+1.62} & \dfrac{7.14s^2+25.8s+9.35}{s^4+20.8s^3+116.4s^2+111.6s+188}\end{bmatrix}$$

绘制出带有 Gershgorin 带的逆 Nyquist 曲线，并在该曲线上标出各个频率下的特征值，验证这些特征值满足 Gershgorin 定理，并绘制该系统的阶跃响应曲线来演示结果系统是不是较好解耦的系统。

20 考虑下面给出的多变量系统，试求出该系统的零点和极点，并判定系统的稳定性。

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -3 & 1 & 2 & 1 \\ 0 & -4 & -2 & -1 \\ 1 & 2 & -1 & 1 \\ -1 & -1 & 1 & -2 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 3 \\ 1 & 1 \end{bmatrix} \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \begin{bmatrix} 1 & 2 & 2 & -1 \\ 2 & 1 & -1 & 2 \end{bmatrix} \boldsymbol{x}(t) \end{cases}$$

注意，多变量系统零点的概念和单变量系统不同，不能由单独求每个子传递函数零点的方式求取，应该由 `tzero()` 函数得出，另外，`pzmap()` 函数同样适用于多变量系统。

21 由控制系统传递函数模型 $G(s) = \dfrac{0.2(s+2)}{s(s+0.5)(s+0.8)(s+3)+0.2(s+2)}$ 写出状态方程实现的可控标准型和可观测标准型。

22 判定下列系统的可控性和可观测性，得出系统的可控和可观测阶梯形式。

① $\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 1 & -3 & 3 & 3 \\ -5 & -1 & -5 & 5 \\ -2 & 0 & -4 & 0 \\ -2 & 0 & -2 & 4 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} u(t),\ \ y(t) = [1, 2, 1, -2]\boldsymbol{x}(t)$

② $\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 1 & -2 & -1 \\ 1 & -2 & -2 \\ -1 & 1 & 2 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} u(t),\ \ y(t) = [1, -1, 0]\boldsymbol{x}(t)$

23 试判定下面矩阵是否为正定矩阵，如果是，则得出其 Cholesky 分解矩阵。

$$\boldsymbol{A} = \begin{bmatrix} 9 & 2 & 1 & 2 & 2 \\ 2 & 4 & 3 & 3 & 3 \\ 1 & 3 & 7 & 3 & 4 \\ 2 & 3 & 3 & 5 & 4 \\ 2 & 3 & 4 & 4 & 5 \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} 16 & 17 & 9 & 12 & 12 \\ 17 & 12 & 12 & 2 & 18 \\ 9 & 12 & 18 & 7 & 13 \\ 12 & 2 & 7 & 18 & 12 \\ 12 & 18 & 13 & 12 & 10 \end{bmatrix}$$

24 试对矩阵 $\boldsymbol{A} = \begin{bmatrix} -2 & 0.5 & -0.5 & 0.5 \\ 0 & -1.5 & 0.5 & -0.5 \\ 2 & 0.5 & -4.5 & 0.5 \\ 2 & 1 & -2 & -2 \end{bmatrix}$ 进行 Jordan 变换，并得出变换矩阵。

25 试求下面齐次方程的基础解系。

$$\begin{cases} 6x_1 + x_2 + 4x_3 - 7x_4 - 3x_5 = 0 \\ -2x_1 - 7x_2 - 8x_3 + 6x_4 = 0 \\ -4x_1 + 5x_2 + x_3 - 6x_4 + 8x_5 = 0 \\ -34x_1 + 36x_2 + 9x_3 - 21x_4 + 49x_5 = 0 \\ -26x_1 - 12x_2 - 27x_3 + 27x_4 + 17x_5 = 0 \end{cases}$$

26 试判定下面的线性代数方程是否有解。

$$\begin{bmatrix}16 & 2 & 3 & 13\\5 & 11 & 10 & 8\\9 & 7 & 6 & 12\\4 & 14 & 15 & 1\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1\\3\\4\\7\end{bmatrix}$$

27 试求出线性代数方程的解析解，并验证解的正确性。

$$\begin{bmatrix}2 & 9 & 4 & 12 & 5 & 8 & 6\\12 & 2 & 8 & 7 & 3 & 3 & 7\\3 & 0 & 3 & 5 & 7 & 5 & 10\\3 & 11 & 6 & 6 & 9 & 9 & 1\\11 & 2 & 1 & 4 & 6 & 8 & 7\\5 & -18 & 1 & -9 & 11 & -1 & 18\\26 & -27 & -1 & 0 & -15 & -13 & 18\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1 & 9\\5 & 12\\4 & 12\\10 & 9\\0 & 5\\10 & 18\\-20 & 2\end{bmatrix}$$

28 试用数值方法和解析方法求取下面的 Sylvester 方程，并验证得出的结果。

$$\begin{bmatrix}3 & -6 & -4 & 0 & 5\\1 & 4 & 2 & -2 & 4\\-6 & 3 & -6 & 7 & 3\\-13 & 10 & 0 & -11 & 0\\0 & 4 & 0 & 3 & 4\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}3 & -2 & 1\\-2 & -9 & 2\\-2 & -1 & 9\end{bmatrix}=\begin{bmatrix}-2 & 1 & -1\\4 & 1 & 2\\5 & -6 & 1\\6 & -4 & -4\\-6 & 6 & -3\end{bmatrix}$$

29 假设某 Riccati 方程的数学表达式为 $\boldsymbol{PA}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}-\boldsymbol{PBR}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}+\boldsymbol{Q}=\boldsymbol{0}$，且

$$\boldsymbol{A}=\begin{bmatrix}-27 & 6 & -3 & 9\\2 & -6 & -2 & -6\\-5 & 0 & -5 & -2\\10 & 3 & 4 & -11\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0 & 3\\16 & 4\\-7 & 4\\9 & 6\end{bmatrix},\ \boldsymbol{Q}=\begin{bmatrix}6 & 5 & 3 & 4\\5 & 6 & 3 & 4\\3 & 3 & 6 & 2\\4 & 4 & 2 & 6\end{bmatrix},\ \boldsymbol{R}=\begin{bmatrix}4 & 1\\1 & 5\end{bmatrix}$$

试求解该方程，得出 $\boldsymbol{P}$ 矩阵，并检验得出解的精度。

30 双输入双输出系统的状态方程表示为

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}2.25 & -5 & -1.25 & -0.5\\2.25 & -4.25 & -1.25 & -0.25\\0.25 & -0.5 & -1.25 & -1\\1.25 & -1.75 & -0.25 & -0.75\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}4 & 6\\2 & 4\\2 & 2\\0 & 2\end{bmatrix}\boldsymbol{u}(t),\boldsymbol{y}(t)=\begin{bmatrix}0 & 0 & 0 & 1\\0 & 2 & 0 & 2\end{bmatrix}\boldsymbol{x}(t)$$

假设选择加权矩阵 $\boldsymbol{Q}=\mathrm{diag}([1,4,3,2])$，且 $\boldsymbol{R}=\boldsymbol{I}_2$，试设计出线性二次型最优调节器，并绘制系统的阶跃响应曲线。如果想改善闭环系统性能，应该如何修改 $\boldsymbol{Q}$ 矩阵？

31　假设已知某 Jordan 块矩阵 $\boldsymbol{A}$ 及其组成部分为

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_1 & & \\ & \boldsymbol{A}_2 & \\ & & \boldsymbol{A}_3\end{bmatrix},\ \boldsymbol{A}_1=\begin{bmatrix}-3 & 1 & 0\\ 0 & -3 & 1\\ 0 & 0 & -3\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}-5 & 1\\ 0 & -5\end{bmatrix},\ \boldsymbol{A}_3=\begin{bmatrix}-1 & 1 & 0 & 0\\ 0 & -1 & 1 & 0\\ 0 & 0 & -1 & 1\\ 0 & 0 & 0 & -1\end{bmatrix}$$

试用解析解运算的方式得出 $\mathrm{e}^{\boldsymbol{A}t}$, $\sin\left(2\boldsymbol{A}t+\dfrac{\pi}{3}\right)$, $\mathrm{e}^{\boldsymbol{A}^2t}\boldsymbol{A}^2+\sin(\boldsymbol{A}^3t)\boldsymbol{A}t+\mathrm{e}^{\sin\boldsymbol{A}t}$。

32　假设已知矩阵 $\boldsymbol{A}$ 如下，试求出 $\mathrm{e}^{\boldsymbol{A}t},\sin\boldsymbol{A}t,\mathrm{e}^{\boldsymbol{A}t}\sin\left(\boldsymbol{A}^2\mathrm{e}^{\boldsymbol{A}t}t\right)$。

$$\boldsymbol{A}=\begin{bmatrix}-4.5 & 0 & 0.5 & -1.5\\ -0.5 & -4 & 0.5 & -0.5\\ 1.5 & 1 & -2.5 & 1.5\\ 0 & -1 & -1 & -3\end{bmatrix}$$

33　请求出下面自治系统状态方程的解析解，并和数值解得出的曲线比较。

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}-5 & 2 & 0 & 0\\ 0 & -4 & 0 & 0\\ -3 & 2 & -4 & -1\\ -3 & 2 & 0 & -4\end{bmatrix}\boldsymbol{x}(t),\quad \boldsymbol{x}(0)=\begin{bmatrix}1\\ 2\\ 0\\ 1\end{bmatrix}$$

34　双输入双输出系统的状态方程表示为

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}2.25 & -5 & -1.25 & -0.5\\ 2.25 & -4.25 & -1.25 & -0.25\\ 0.25 & -0.5 & -1.25 & -1\\ 1.25 & -1.75 & -0.25 & -0.75\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}4 & 6\\ 2 & 4\\ 2 & 2\\ 0 & 2\end{bmatrix}\boldsymbol{u}(t),\ \boldsymbol{y}(t)=\begin{bmatrix}0 & 0 & 0 & 1\\ 0 & 2 & 0 & 2\end{bmatrix}\boldsymbol{x}(t)$$

试将该模型输入到 MATLAB 空间，并得出该模型相应的传递函数矩阵。若选择采样周期为 $T=0.1$ 秒，求出离散化后的状态方程模型和传递函数矩阵模型。对该模型进行连续化变换，测试一下能否变换回原来的模型。

35　给出一个 8 阶系统模型 $G(s)$

$$\frac{18s^7+514s^6+5982s^5+36380s^4+122664s^3+222088s^2+185760s+40320}{s^8+36s^7+546s^6+4536s^5+22449s^4+67284s^3+118124s^2+109584s+40320}$$

并假定系统具有零初始状态，请求出单位阶跃响应和脉冲响应的解析解。若输入信号变为正弦信号 $u(t)=\sin(3t+5)$，请求出零初始状态下系统时域响应的解析解，并用图形的方法进行描述，和数值解进行比较。

参考文献

[1] Laub A. Numerical linear algebra aspects of control design computations [J]. IEEE Transactions on Automatic Control, 1985, AC-30(2):97～108

[2] 黄琳. 系统与控制理论中的线性代数 [M]. 北京: 科学出版社, 1984

[3] The MathWorks Inc. Statistics toolbox user's manual [Z], 2004

[4] 薛定宇，陈阳泉. 高等应用数学问题的 MATLAB 求解 [M]. 北京: 清华大学出版社, 2004

[5] Dongarra J J, Bunsh J R, Molor C B. LINPACK user's guide [M]. Philadelphia: SIAM Press, 1979

[6] Press W H, Flannery B P, Teukolsky S A, et al. Numerical recipes, the art of scientific computing [M]. Cambridge: Cambridge University Press, 1986

[7] Garbow B S, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide extension [M], Lecture notes in computer sciences, volume 51. New York: Springer-Verlag, 1977

[8] Smith B T, Boyle J M, Dongarra J J, et al. Matrix eigensystem routines — EISPACK guide [M], Lecture notes in computer sciences, volume 6. New York: Springer-Verlag, second edition , 1976

[9] Moler C B, Stewar G W. An algorithm for generalized matrix eigenvalue problems [J]. SIAM Journal of Numerical Analysis, 1973, 10(241~256)

[10] 韩京清，何关钰，许可康. 线性系统理论代数基础 [M]. 沈阳: 辽宁科学技术出版社, 1985

[11] Kalman R. On the general theory of control systems [J]. IRE Transactions on Automatic Control, 1959, 4(3):110. Abstract. Full paper published on the 1st IFAC Congress, Moscow, 1960

[12] Kuo B C. Digital control systems [M]. New York: Holt, Rinehart and Winston. Inc, 1980

[13] Rosenbrock H H. Computer-aided control system design [M]. New York: Academic Press, 1974

[14] 肖筱南. 现代数值计算方法 [M]. 北京: 北京大学出版社, 2003

[15] Klema V, Laub A. The singular value decomposition: Its computation and some applications [J]. IEEE Transactions on Automatic Control, 1980, AC-25(2):164~176

[16] 郑大钟. 线性系统理论 [M]. 北京: 清华大学出版社, 1980

[17] Moore B. Principal component analysis in linear systems: Controllability, observability, and model reduction [J]. IEEE Transactions on Automatic Control, 1981, AC-26(1):17~32

[18] 《数学手册》编写组. 数学手册 [M]. 北京: 人民教育出版社, 1979

[19] Laub A. A Schur method for solving algebraic Riccati equations [J]. IEEE Transactions on Automatic Control. 1979. AC-24(6):913~921

[20] Arnold W F III, Laub A. Generalized eigenproblem algorithms and software for algebraic Riccati equations [J]. Proceedings of IEEE, 1984, 72(12):1746~1754

[21] Anderson B D O, Moore J B. Linear optimal control [M]. Englewood Cliffs: Prentice-Hall, 1971

[22] Franklin G F, Powell J D, Workman M. Digital control of dynamic systems [M]. Reading MA: Addison Wesley, 3rd , 1988. (清华大学出版社有影印版)

[23] Moler C B, Van Loan C F. Nineteen dubious ways to compute the exponential of a matrix [J]. SIAM Review, 1979, 20:801~836

[24] 孙增圻，袁曾任. 控制系统的计算机辅助设计 [M]. 北京: 清华大学出版社, 1988

[25] 薛定宇，任兴权. 连续系统的仿真与解析解法 [J]. 自动化学报, 1992, 19(6): 694~702

第 4 章 常微分方程问题的计算机求解

微分方程是描述动态系统的最常用数学工具，也是很多科学与工程领域数学建模的基础。线性微分方程和低阶特殊微分方程往往可以通过解析解的方法求解，但一般的非线性微分方程是没有解析解的，故需要用数值解的方式求解。4.1 节对线性常系数微分方程的解析解进行研究，给出基于 MATLAB 语言的微分方程解析解方法，探讨了简单非线性系统的解析解的可能性，并通过几个实际的物理建模实例介绍系统微分方程的建模方法与求解方法。由于绝大多数非线性微分方程的解析解是不存在的，所以从 4.2 节开始主要介绍了一般微分方程的数值解法。4.2 节首先介绍一阶显式常微分方程组的数值求解方法和关键技术，介绍了几种常用的微分方程数值解算法，引入变步长算法的概念与必要性，以及基于 MATLAB 现成函数的微分方程求解方法，还通过例子对微分方程数值解过程中遇到的问题和采用的方法进行阐述，并介绍一些求解技巧，包括微分方程的 MATLAB 描述方法、变步长的相对误差准则选取、一般微分方程转换成一阶显式微分方程的方法等。4.3 节研究各种其他类型微分方程初值问题的求解，包括刚性微分方程的数值求解方法、隐式微分方程的求解方法、微分代数方程的求解方法、延迟微分方程和切换微分方程的求解方法等，这些微分方程的求解在控制系统研究中均能找到其对应的原型问题。4.4 节研究微分方程的边值问题计算机求解方法，首先介绍二阶微分方程的求解方法，再介绍一般边值问题的求解方法及扩展边值问题的求解方法，并根据该方程的求解方法引入微分 Riccati 方程的求解方法，并介绍二次型最优控制问题的求解方法。在 4.5 节中将简要介绍基于 Simulink 的系统微分方程建模与求解方法，并以多变量系统、计算机控制系统、时变系统和网络控制系统为例，研究基于框图的建模方法与仿真方法。

4.1 常系数线性微分方程的解析解方法

4.1.1 微分方程的解析解方法

MATLAB 语言的符号运算工具箱提供了一个线性常系数微分方程求解的实用函数 dsolve()，该函数允许用字符串的形式描述微分方程及初值、边值条件，最终将得出微分方程的解析解。该函数的调用格式为

```
y=dsolve(f1, f2, ..., fm)
y=dsolve(f1, f2, ..., fm,'x')      % 指明自变量
```

其中，f_i 既可以描述微分方程，又可以描述初始条件或边界条件。在描述微分方程时，可以用 D4y 这样的记号表示 $y^{(4)}(t)$，还可以用 D2y(2)=3 这类记号表示 $\ddot{y}(2)=3$ 这样的已知条件，该函数可以容易地得出原微分方程的解。如果描述微分方程的自变量不是 t 而是 x，则可以由后一个 MATLAB 语句格式指明自变量。

例 4-1 考虑图 2-3 中的 RLC 串联电路，其对应的微分方程模型为

$$LC\frac{\mathrm{d}^2u_\mathrm{c}(t)}{\mathrm{d}t^2}+RC\frac{\mathrm{d}u_\mathrm{c}(s)}{\mathrm{d}t}+u_\mathrm{c}(t)=u(t) \tag{4-1-1}$$

试求该系统在零初始条件下阶跃响应的解析解 $u_\mathrm{c}(t)$。

求解 用下面的语句可以立即得出阶跃响应解析解为

```
>> syms L R C
   uc=dsolve('L*C*D2Uc+R*C*DUc+Uc=1','Uc(0)=0','DUc(0)=0')
```

这样可以得出系统的阶跃响应解析解为

$$u_\mathrm{c}(t)=-\frac{RC+\Delta}{2\Delta}\mathrm{e}^{-\frac{(RC-\Delta)t}{2LC}}+\frac{RC-\Delta}{2\Delta}\mathrm{e}^{-\frac{(RC+\Delta)t}{2LC}}+1$$

其中 $\Delta=\sqrt{R^2C^2-4LC}$。

例 4-2 假设输入信号为 $u(t)=\mathrm{e}^{-5t}\cos(2t+1)+5$，试求出下面微分方程的通解。

$$y^{(4)}(t)+10y^{(3)}(t)+35\ddot{y}(t)+50\dot{y}(t)+24y(t)=5\ddot{u}(t)+4\dot{u}(t)+2u(t)$$

求解 若想求解本微分方程，首先应该定义 t 为符号变量，这样就可以推导出给定微分方程等式右侧的时间表达式为

```
>> syms t y; u=exp(-5*t)*cos(2*t+1)+5;
   uu=5*diff(u,t,2)+4*diff(u,t)+2*u;
   y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)])
```

上述语句中，先由给定的输入信号 $u(t)$ 直接推导出该方程的右侧表达式，得出新的符号变量 uu。因为 dsolve() 函数需要字符串型的变量描述微分方程，所以用

char() 函数将等号右侧的表达式转换成字符串，与等号左侧的表达式一起描述完整的微分方程。这样得出的微分方程解析解为

$$y(t)=\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)+C_1\mathrm{e}^{-4t}+C_2\mathrm{e}^{-3t}+C_3\mathrm{e}^{-2t}+C_4\mathrm{e}^{-t}$$

其中，C_i 为任意常数。若给出初始条件或边界条件，则可以通过这些条件建立方程，求出 C_i 的值。这样的结果和高等数学中微分方程求解是一致的。

仍考虑上面的微分方程，假设已知 $y(0)=3, \dot{y}(0)=2, \ddot{y}(0)=y^{(3)}(0)=0$，则可以通过下面的命令求取满足该微分方程的特解。

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)],...
      'y(0)=3','Dy(0)=2','D2y(0)=0','D3y(0)=0')
```

由于得出的解较冗长，所以这里只给出自动转换的 LaTeX 结果如下：

$$\begin{aligned}y(t)=&\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)\\&+\left(-\frac{445}{26}\cos 1-\frac{51}{13}\sin 1-\frac{69}{2}\right)\mathrm{e}^{-2t}+\left(-\frac{271}{30}\cos 1+\frac{41}{15}\sin 1-\frac{25}{4}\right)\mathrm{e}^{-4t}\\&+\left(\frac{179}{8}\cos 1+\frac{5}{8}\sin 1+\frac{73}{3}\right)\mathrm{e}^{-3t}+\left(\frac{133}{30}\cos 1+\frac{97}{60}\sin 1+19\right)\mathrm{e}^{-t}\end{aligned}$$

利用强大的 MATLAB 符号运算工具箱，还可以求解出以前看似不可能的问题的解析解。例如，设置 $y(0)=1/2, \dot{y}(\pi)=1, \ddot{y}(2\pi)=0, \dot{y}(2\pi)=1/5$，则可以得出解析解为

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',char(uu)],...
      'y(0)=1/2','Dy(pi)=1','D2y(2*pi)=0','Dy(2*pi)=1/5')
```

如果用推导的方法求 C_i 的值，则每个系数的解析解至少要写出 10 数行，所以应该采用有理式近似的方式 vpa(y,10) 将方程的解析解表示成

$$\begin{aligned}y(t)\approx&\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)\\&-219.1291604\mathrm{e}^{-t}+442590.9052\mathrm{e}^{-4t}+31319.63786\mathrm{e}^{-2t}-473690.0889\mathrm{e}^{-3t}\end{aligned}$$

从表面上看，这里的微分方程求解问题似乎和前面介绍的传递函数的时域响应解析解好像是同样的问题，事实上，它们之间是有区别的。因为传递函数是建立在系统输入、输出信号及其导数初值均为 0 的前提下的，所以以前介绍的传递函数方法不适合于求解上述微分方程的解析解问题。

例 4-3 前面介绍的方程只含有实数极点，其实符号运算工具箱提供的 dsolve() 函数同样适用于有复数极点的微分方程解析解。假设微分方程如下

$$y^{(5)}(t)+5y^{(4)}(t)+12y^{(3)}(t)+16\ddot{y}(t)+12\dot{y}(t)+4y(t)=\dot{u}(t)+3u(t)$$

且假设输入信号为正弦信号 $u(t)=\sin t$，并假设 $y(0)=\dot{y}(0)=\ddot{y}(0)=y^{(3)}(0)=y^{(4)}(0)=0$，试用解析方法求解该方程。

求解　用下面的方法可以求出原微分方程的解析解

```
>> syms t y; u=sin(t); uu=3*diff(u)+3*u;
   y=dsolve(['D5y+5*D4y+12*D3y+16*D2y+12*Dy+4*y=' char(uu)],...
     'y(0)=0','Dy(0)=0','D2y(0)=0','D3y(0)=0','D4y(0)=0')
```

其解析解的数学描述为

$$y(t)=-\frac{12}{25}\cos t-\frac{9}{25}\sin t+\frac{57}{50}\mathrm{e}^{-t}\sin t+\frac{12}{25}\mathrm{e}^{-t}\cos t+\frac{3}{5}t\mathrm{e}^{-t}\sin t-\frac{3}{10}t\mathrm{e}^{-t}\cos t$$

更简单地，解析解可以手工修改为

$$y(t)=-\frac{12}{25}\cos t-\frac{9}{25}\sin t+\left(\frac{57}{50}+\frac{3}{5}t\right)\mathrm{e}^{-t}\sin t+\left(\frac{12}{25}-\frac{3}{10}t\right)\mathrm{e}^{-t}\cos t$$

例 4-4　试求线性微分方程组 $\begin{cases}\ddot{x}(t)+2\dot{x}(t)=x(t)+2y(t)-\mathrm{e}^{-t}\\ \dot{y}(t)=4x(t)+3y(t)+4\mathrm{e}^{-t}\end{cases}$ 的解析解。

求解　线性微分方程组也可以用 dsolve() 函数直接求解。上述的线性微分方程组可以由下面的 MATLAB 语句直接求解。

```
>> [x,y]=dsolve('D2x+2*Dx=x+2*y-exp(-t)','Dy=4*x+3*y+4*exp(-t)')
```

这样可以求出方程的解析解为

$$\begin{cases}x(t)=-6t\mathrm{e}^{-t}+C_1\mathrm{e}^{-t}+C_2\mathrm{e}^{(1+\sqrt{6})t}+C_3\mathrm{e}^{-(-1+\sqrt{6})t}\\ y(t)=6t\mathrm{e}^{-t}-C_1\mathrm{e}^{-t}+2\left(2+\sqrt{6}\right)C_2\mathrm{e}^{(1+\sqrt{6})t}+2\left(2-\sqrt{6}\right)C_3\mathrm{e}^{-(-1+\sqrt{6})t}+\dfrac{1}{2}\mathrm{e}^{-t}\end{cases}$$

4.1.2　特殊非线性微分方程的解析解

特殊的非线性微分方程也是可以用 dsolve() 函数求解析解的，这样的方程描述方式和前面介绍的线性微分方程是一致的，描述了这样的微分方程，则可以直接求解出微分方程的解析解。下面通过例子演示非线性方程的解析解求解，同时还将演示不能求解的例子。

例 4-5　试求出一阶非线性微分方程 $\dot{x}(t)=x(t)[1-x^2(t)]$ 的解析解。

求解　这样简单的一阶非线性方程可以考虑用 dsolve() 函数直接解出。

```
>> syms t x; x=dsolve('Dx=x*(1-x^2)')
```

该微分方程的解析解为 $x(t)=\pm 1/\sqrt{1+C_1\mathrm{e}^{-2t}}$。

其实，稍微改变原微分方程，例如将等号右侧加上 1，则可以用下面的语句试解该方程。读者会发现原始的微分方程是没有解析解的。

```
>> syms t x;  x=dsolve('Dx=x*(1-x^2)+1')
```

这时出现警告信息“Explicit solution could not be found”，说明原方程不存在解析解。

例 4-6 考虑著名的 Van der Pol 方程 $\dfrac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+\mu[y^2(t)-1]\dfrac{\mathrm{d}y(t)}{\mathrm{d}t}+y(t)=0$，试用 dsolve() 函数求解它，看看能得出什么结论。

求解 由前面的讨论可见，似乎所有的微分方程都可以直接用 MATLAB 语言提供的强大的 dsolve() 函数求解，这样很自然地想到一般非线性微分方程的解析解问题。

对前面给出的 Van der Pol 方程，用户尝试如下的 MATLAB 命令，也将得出原微分方程无解析解的提示。

```
>> syms y mu; y=dsolve('D2y+mu*(y^2-1)*Dy+y')
```

这时将给出警告信息，表示原方程没有解析解。

可见，微分方程解析解求解函数 dsolve() 并不能直接应用于一般非线性方程的解析解的求解，因为它们的解析解确实不存在。所以非线性微分方程只能用数值解法去求解，即使看起来很简单的非线性微分方程也是没有解析解的，只有极特殊的非线性微分方程解析可解。下面将集中介绍各类非线性微分方程的数值解方法。

4.2 微分方程问题的数值解法

前面介绍了微分方程的解析解方法，同时也指出很多非线性微分方程是不存在解析解法的，需要使用数值解法对之进行研究。从本节开始着重讨论基于 MATLAB/Simulink 语言的各类微分方程的数值解方法。

4.2.1 微分方程问题算法概述

一般微分方程的数值解法很大一类是关于微分方程初值问题的数值解法，这类问题需要用一阶显式的微分方程组来描述为

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(t,\boldsymbol{x}(t)) \tag{4-2-1}$$

其中，$\boldsymbol{x}^{\mathrm{T}}(t)=[x_1(t),x_2(t),\cdots,x_n(t)]$ 称为状态向量，方程左侧是状态变量的一阶导数向量。右侧的函数 $\boldsymbol{f}^{\mathrm{T}}(\cdot)=[f_1(\cdot),f_2(\cdot),\cdots,f_n(\cdot)]$ 可以是任意非线性函数。所谓初值问题是指，若已知初始状态 $\boldsymbol{x}_0=[x_1(0),\cdots,x_n(0)]^{\mathrm{T}}$，用数值求解方法求出在某个时间区间 $t\in[0,t_{\mathrm{f}}]$ 内各个时刻状态变量 $\boldsymbol{x}(t)$ 的数值，这里 t_{f} 又称为终止时间。

对多元非线性常微分方程初值问题来说，Euler 算法是最直观的一类求解算法。虽然该算法比较简单，但对理解其他复杂的微分方程算法是很有帮助的，故这里将以 Euler 算法为例介绍微分方程初值问题的数值算法。

假设已知在 t_0 时刻系统状态向量的值为 $\boldsymbol{x}(t_0)$，若选择一个很小的计算步长 h，则可以将微分方程左侧的导数近似为 $(\boldsymbol{x}(t_0+h)-\boldsymbol{x}(t_0))/(t_0+h-t_0)$，代入微分方程则可以解出在 t_0+h 时刻方程的近似解为

$$\hat{\boldsymbol{x}}(t_0+h)=\boldsymbol{x}(t_0)+h\boldsymbol{f}(t_0,\boldsymbol{x}(t_0)) \tag{4-2-2}$$

更严格地，因为这样的近似解是存在误差的，所以可以写出在 t_0+h 时刻系统状态向量的值为

$$\boldsymbol{x}(t_0+h)=\hat{\boldsymbol{x}}(t_0+h)+\boldsymbol{R}_0=\boldsymbol{x}_0+h\boldsymbol{f}(t,\boldsymbol{x}_0)+\boldsymbol{R}_0 \tag{4-2-3}$$

简记 $\boldsymbol{x}_1=\boldsymbol{x}(t_0+h)$，则 $\hat{\boldsymbol{x}}_1=\hat{\boldsymbol{x}}(t_0+h)$ 为系统状态向量在 t_0+h 时刻的近似值，亦即数值解。可见，$\boldsymbol{R}_0$ 为数值解的舍入误差。在实际解法中为简单起见，经常可以舍弃 ^ 记号，而将数值解直接记为 $\boldsymbol{x}_1$。

一般地，假设已知在 t_k 时刻系统的状态向量为 $\boldsymbol{x}_k$，则在 t_k+h 时刻 Euler 算法的数值解可以写成

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+h\boldsymbol{f}(t,\boldsymbol{x}_k) \tag{4-2-4}$$

这样，用迭代的方法可以由给定的初值问题逐步求出在所选择的时间段 $t\in[0,T]$ 内各个时刻 $t_0+h,t_0+2h,\cdots$ 处的原问题数值解。

提高数值解精度的一种方法是减小步长 h 的值。然而，并不能无限制地减小 h 的值，这主要有两个原因：

① **减慢计算速度** 因为对选定的求解时间而言，减小步长就意味着增加在这个时间段内的计算点数目，故计算速度减慢。

② **增加累积误差** 因为不论选择多小的步长，所得出的数值解都将有一个舍入误差，减小计算步长则将增加计算的次数，从而使得整个计算过程的舍入误差的叠加和传递次数增多，产生较大的累积误差。舍入误差、累积误差和总误差关系的示意图如图 4-1 所示。

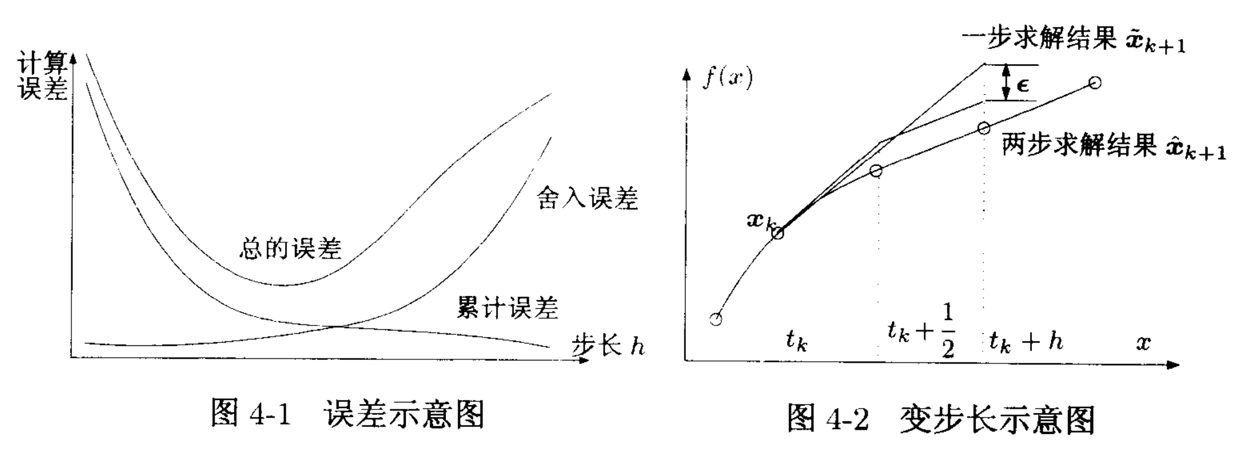


图 4-1 误差示意图

图 4-2 变步长示意图

所以在对微分方程求解过程中，应采取下列措施：

① **选择适当的步长** 采用像 Euler 法这样简单的算法时，应适当地选择步长，既不能太大，又不能太小。

② **改进近似算法精度** 由于 Euler 算法只是将原积分问题进行梯形的近似，其近似精度很低，从而不能很有效地逼近原始问题。可以用各种更精确的插值方法来取代 Euler 算法，从而改进运算精度。比较成功的是 Runge-Kutta 法、Adams 法等。

③ **采用变步长方法** 前面提及“适当”地选择步长，这本身就是个模糊的概念，如何适当地选择步长取决于经验。事实上，很多种方法都允许变步长的求解，如果误差较小时，可自动地增大步长，而误差较大时再自动减小步长，从而精确、有效地求解给出的常微分方程初值问题。

一般变步长算法的原理如图 4-2 所示。已知 t_k 时刻的状态变量 $\boldsymbol{x}_k$，则先在某步长 h 下计算出 t_k+h 时刻的状态变量 $\tilde{\boldsymbol{x}}_{k+1}$。另外，将步长变成原来步长的一半，分两步从 $\boldsymbol{x}_k$ 计算出 t_k+h 时刻的状态变量 $\hat{\boldsymbol{x}}_{k+1}$。如果两种运算步长下的误差 $\epsilon=|\hat{\boldsymbol{x}}_{k+1}-\tilde{\boldsymbol{x}}_{k+1}|$ 小于给定的误差限，则可以采用该步长或适当增大步长，如果误差大，则进一步减小步长。自适应变步长算法可以较好地解决计算速度问题，另外能保证计算的精度。

4.2.2 四阶定步长 Runge-Kutta 算法及 MATLAB 实现

四阶定步长的 Runge-Kutta 算法是传统数值分析课程和系统仿真课程中经常介绍的算法，被认为是求解微分方程的一种有效的方法，该算法结构很简单，可以先定义如下 4 个附加向量：

$$\begin{cases} \boldsymbol{k}_1=h\boldsymbol{f}(t_k,\boldsymbol{x}_k) \\ \boldsymbol{k}_2=h\boldsymbol{f}\left(t_k+\dfrac{h}{2},\boldsymbol{x}_k+\dfrac{\boldsymbol{k}_1}{2}\right) \\ \boldsymbol{k}_3=h\boldsymbol{f}\left(t_k+\dfrac{h}{2},\boldsymbol{x}_k+\dfrac{\boldsymbol{k}_2}{2}\right) \\ \boldsymbol{k}_4=h\boldsymbol{f}(t_k+h,\boldsymbol{x}_k+\boldsymbol{k}_3) \end{cases} \tag{4-2-5}$$

其中，h 为计算步长，在实际应用中该步长是一个常数，这样由四阶 Runge-Kutta 算法可以求解出下一个步长的状态变量值为

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\frac{1}{6}(\boldsymbol{k}_1+2\boldsymbol{k}_2+2\boldsymbol{k}_3+\boldsymbol{k}_4) \tag{4-2-6}$$

这样，用迭代的方法由给定的初值问题逐步求出在所选择的时间段 $t\in[t_0,t_{\rm h}]$ 内各个时刻 $t_0+h,t_0+2h,\cdots$ 处的原问题数值解。

有了上面的数学算法，则可以用 MATLAB 语言编写出该算法的函数如下：

```
function [tout,yout]=rk_4(odefile,ts,y0)
t0=ts(1); th=ts(2); yout=[]; tout=[];
if length(ts)==3, h=ts(3); else, h=(ts(2)-ts(1))/100;th=ts(2);end
for t=[t0:h:th]
   k1=h*feval(odefile,t,y0);
   k2=h*feval(odefile,t+h/2,y0+0.5*k1);
   k3=h*feval(odefile,t+h/2,y0+0.5*k2);
   k4=h*feval(odefile,t+h,y0+k3);
   y0=y0+(k1+2*k2+2*k3+k4)/6; yout=[yout; y0']; tout=[tout; t];
end
```

其中，tspan 可以有两种构成方法，第一种方法可以是一个等间距的时间向量；第二种方法是 tspan=[t_0,t_h,h]，其中，t_0 和 t_h 为计算的初始和终止值，h 为计算步长，odefile 是一个字符串变量，为表示微分方程的文件名，$\boldsymbol{y}_0$ 是初值列向量。函数调用完成后，时间向量与各个时刻状态变量构成的矩阵分别由 tout 和 yout 返回。

该算法看似简单，然而从求解数值问题的效果看，该算法不是一个较好的方法，故一般不使用该方法，后面将通过例子演示微分方程求解方法，并和现成的 MATLAB 函数进行对比分析。

4.2.3 一阶微分方程组的数值解

这里首先给出一种微分方程求解的高精度变步长四阶五级 Runge-Kutta-Felhberg 算法，然后着重介绍如何用 MATLAB 语言的函数直接求解方法与实例，并将介绍非线性微分方程收敛性的仿真研究结果。

1. 四阶五级 Runge-Kutta-Felhberg 算法

德国学者 Felhberg 对传统的 Runge-Kutta 方法进行了改进[1]，在每一个计算步长内对 $f_i(\cdot)$ 函数进行 6 次求值，以保证更高的精度和数值稳定性，该算法又称为四阶五级 RKF 算法。假设当前步长为 h_k，则可以定义下面 6 个 $\boldsymbol{k}_i$ 变量：

$$\boldsymbol{k}_i = h_k \boldsymbol{f}\left(t_k + \alpha_i h_k, \boldsymbol{x}_k + \sum_{j=1}^{i-1} \beta_{ij} \boldsymbol{k}_j\right), \quad i = 1, 2, \cdots, 6 \tag{4-2-7}$$

式中，t_k 为当前计算时刻，中间参数 α_i, β_{ij} 及其他参数由表 4-1 给出，其中，α_i, β_{ij} 参数对又称为 Dormand-Prince 对。这时下一步的状态向量可以由下式求出。

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \sum_{i=1}^{6} \gamma_i \boldsymbol{k}_i \tag{4-2-8}$$

表 4-1 四阶五级 RKF 算法系数表

α_i	β_{ij}					γ_i	γ_i^*
0						16/135	25/216
1/4	1/4					0	0
3/8	3/32	9/32				6656/12825	1408/2565
12/13	1932/2197	−7200/2197	7296/2197			28561/56430	2197/4104
1	439/216	−8	3680/513	−845/4104		−9/50	−1/5
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	2/55	0

当然，直接采用这一方法为定步长的方法。在实际问题中往往希望在一些情况下 (如解变化很快时) 采用较小的步长，而在另一些情况下 (如解的变化很缓慢时) 采用较大的步长，这样做既可以保证较高的精度，又可以保证较高的运算速度。在此算法中，定义一个误差向量 $\epsilon_k = \sum_{i=1}^{6}(\gamma_i - \gamma_i^*)\boldsymbol{k}_i$，可以由它的大小来变换计算步长，所以这种能自动变换步长的方法又称为自适应变步长方法。

定步长算法相当于用开环控制的思想求解微分方程，选定步长后将不考虑计算是不是有误差，也不考虑误差是否能被接受，一味采用单一步长计算全程。而变步长算法则是采用由误差作为监测指标的闭环控制思想，在计算过程中修正步长，使得预期的计算精度得以保证，同时由于采用变步长思想，所以大部分问题的求解时间将明显缩短，因为在计算过程中在保证计算精度的前提下也允许大步长。

2. 基于 MATLAB 的微分方程求解函数

MATLAB 下求解一阶微分方程组初值问题数值解的最常用的方法是 `ode45()` 函数，该函数实现了前面介绍的变步长四阶五级 RKF 算法，可以采用变步长的算法求解微分方程。该函数的调用格式为

```
[t,x]=ode45(Fun,[t0,tf],x0)                          % 直接求解
[t,x]=ode45(Fun,[t0,tf],x0,options)                  % 带有控制选项
[t,x]=ode45(Fun,[t0,tf],x0,options,p1,p2,···)        % 带有附加参数
```

其中，微分方程应该用 MATLAB 函数 `Fun`、`inline()` 或匿名函数按指定的格式描述，有关该函数的编写方法后面将通过例子专门介绍，$[t_0, t_f]$ 描述微分方程求解的区间，如果只给出一个值 t_f 则表示初始时刻为 $t_0 = 0$，终止时刻为 t_f 的问题求解。为使得微分方程能够求解，还应该已知初值问题的初始状态变量 $\boldsymbol{x}_0$。另外，该函数还允许 $t_f < t_0$，即可以认为 t_0 为终值时刻，t_f 为初始时刻，故 $\boldsymbol{x}_0$ 表示状态变量的终值，故该函数可以直接求解终值问题。

求解一阶显式微分方程组的关键是用 MATLAB 语言编写一个函数，描述需

要求解的微分方程组。该函数的入口应该为[1]

```
function xd=funname(t,x)                     % 不需附加变量的格式
function xd=funname(t,x,flag,p1,p2,···)      % 可以使用附加变量
```

其中，t 是时间变量或自变量，即使需要求解的微分方程不是时变的，也需要给出 t 占位，否则在变量传递过程中将出现问题。变量 $\boldsymbol{x}$ 为状态向量，返回的 $\boldsymbol{x}_\mathrm{d}$ 为状态向量的导数。flag 一般用来控制求解过程，可以用其给初始状态赋值。

在微分方程求解中有时需要对求解算法及控制条件进行进一步设置，这可以通过求解过程中的 options 变量进行修改。初始 options 变量可以通过 odeset() 函数获取，该变量是一个结构体变量，其中有众多成员变量，表 4-2 中列出了常用的一些成员变量。

表 4-2　常微分方程求解函数的控制参数表

参数名	参 数 说 明
RelTol	为相对误差容许上限，默认值为 0.001 (即 0.1% 的相对误差)，在一些特殊的微分方程求解中，为了保证较高的精度，还应该再适当减小该值，如 1e-8 或 1e-6
AbsTol	为一个向量，其分量表示每个状态变量允许的绝对误差，其默认值为 10^{-6}。当然可以自由设置其值，以改变求解精度
MaxStep	为求解方程最大允许的步长
Mass	微分代数方程中的质量矩阵，可用于描述微分代数方程
Jacobian	为描述 Jacobi 矩阵函数 $\partial f/\partial x$ 的函数名，如果已知该 Jacobi 矩阵，则能加速仿真过程

修改该变量有两种方式，其一是用 odeset() 函数设置，其二是直接修改 options 的成员变量。例如，若想将相对误差设置成较小的 10^{-7}，则需要给出

```
options=odeset('RelTol',1e-7);          % 或更直观地
options=odeset; options.RelTol=1e-7;
```

在实际求解过程中经常需要定义一些附加参数，这些参数由 $p_1,p_2,\cdots,p_m$ 表示，在编写方程函数时也应该一一对应地写出，在这种调用格式下还应该使用 flag 变量占位。后面将通过例子详细介绍相关的调用格式。

例 4-7 考虑著名的 Rössler 方程，其数学表达式为

$$\begin{cases}\dot{x}(t)=-y(t)-z(t)\\ \dot{y}(t)=x(t)+ay(t)\\ \dot{z}(t)=b+[x(t)-c]z(t)\end{cases}$$

[1] 在带有附加参数时，匿名函数和 M-函数与 inline() 函数的入口语句是不相同的，后二者在附加参数前还应该有一个 flag 变量占位，而匿名函数不能写这个 flag 变量。

选定 $a=b=0.2$, $c=5.7$, 且 $x(0)=y(0)=z(0)=0$。试仿真该系统，绘制出系统的时域响应和相空间轨迹。

求解 该方程是非线性微分方程，所以不存在解析解，只能用数值解法求解。若想用数值算法求解该微分方程，首先应该引入状态变量向量，如 $x_1=x, x_2=y, x_3=z$，这样，原微分方程可以写成如下的一阶显式微分方程组

$$\begin{cases}\dot{x}_1(t)=-x_2(t)-x_3(t)\\ \dot{x}_2(t)=x_1(t)+ax_2(t)\\ \dot{x}_3(t)=b+[x_1(t)-c]x_3(t)\end{cases}$$

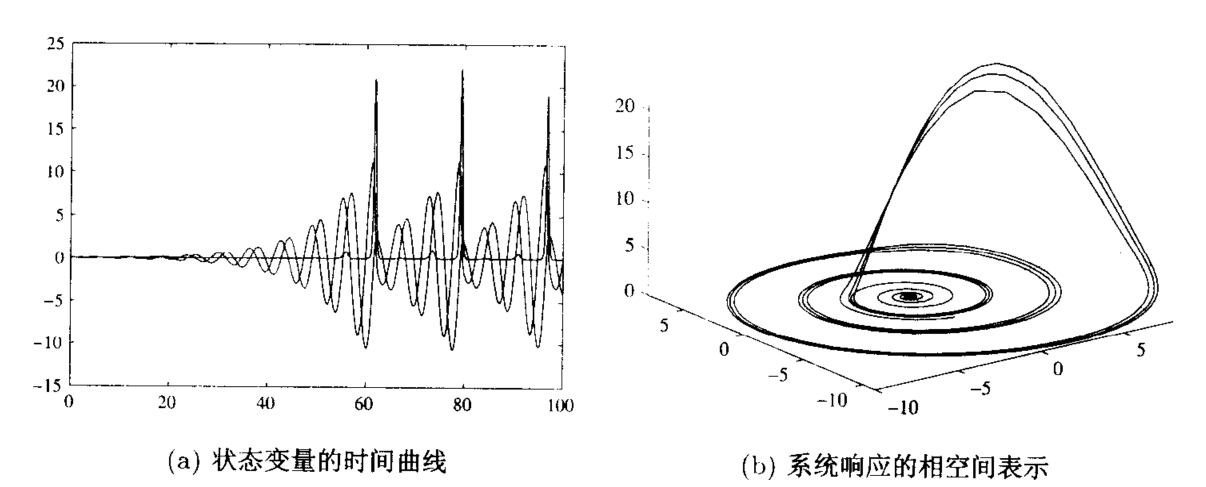
根据这样的方程组，可以按下面的格式编写出一个 MATLAB 函数 rossler.m 来描述上面的微分方程模型。其内容为

```
function dx=rossler(t,x) % 虽然不显含时间，还应该写出占位
dx=[-x(2)-x(3);  % 对应方程第一行，直接将参数代入
    x(1)+0.2*x(2); 0.2+(x(1)-5.7)*x(3)]; % 其余两行
```

这时，可以调用微分方程数值求解函数 ode45() 对 rossler() 函数描述的系统进行数值求解，并将结果进行图形显示。

```
>> t_final=100; x0=[0;0;0]; opt=odeset; opt.RelTol=1e-8;
   [t,x]=ode45(@rossler,[0,t_final],x0,opt); plot(t,x),
   figure; plot3(x(:,1),x(:,2),x(:,3));
```

其中，t_final 为设定的仿真终止时间，x0 为初始状态。第一个绘图命令绘制出系统的各个状态的时间响应曲线图，如图 4-3 (a) 所示。第二个绘图命令可以绘制出三个状态的相空间曲线，如图 4-3 (b) 所示。可以看出，看似很复杂的三元一阶常微分方程组的数值解问题由几条简单直观的 MATLAB 语句就求解出来了。此外，用 MATLAB 语言还可以轻易、直观地将结果用可视化方式直接显示出来。



(a) 状态变量的时间曲线 (b) 系统响应的相空间表示

图 4-3 Rössler 方程的仿真结果

其实，观察相空间轨迹走行的最好方法是采用 comet3() 函数绘制动画式的轨迹，故可将最后一个语句改成 comet3(x(:,1),x(:,2),x(:,3))。

从前面的微分方程求解实例看，如果能建立一个描述微分方程的 MATLAB 函数，则调用 ode45() 将立即得出该微分方程的数值解。可以看出，编写一个 MATLAB 函数来描述微分方程是常微分方程初值问题数值求解的关键。下面将介绍能描述微分方程的另一种方法：匿名函数定义方法。

例 4-8 考虑例 4-7 中给出的 Rössler 方程，试用匿名函数描述该方程，并求出该微分方程的数值解，与前面的结果进行比较。

求解 可以用 rossler() 函数或用下面的语句直接定义该方程。

```
>> f1=@(t,x)[-x(2)-x(3); x(1)+0.2*x(2); 0.2+(x(1)-5.7)*x(3)];
```

其中，f1 为生成的函数名，可以用 ode45() 这类微分方程求解函数直接调用。这里自变量仍为 t 和 x，微分方程的函数内容一行显示不下，故将其拆成两行显示。定义了微分方程模型，就可以通过下面的语句求解这个微分方程了。采用下面的命令，将得出与例 4-7 完全一致的结果。

```
>> t_final=100; x0=[0;0;0]; [t,x]=ode45(f1,[0,t_final],x0,opt);
   plot(t,x), figure; plot3(x(:,1),x(:,2),x(:,3));
```

3. MATLAB 下带有附加参数的微分方程求解

在基于 MATLAB 语言的微分方程求解中，引入附加参数的目的是，若微分方程的某些参数可以选择不同的值，对不同值求解时考虑附加参数可以避免每次修改模型文件。例如，例 4-7 中给出的 Rössler 方程中，a,b,c 可以看成附加参数，这样在表示它们的变化时，无需修改描述原始微分方程的 MATLAB 函数，而只需在调用微分方程求解时从外部修改它们的参数即可。

例 4-9 试编写带有附加参数的 MATLAB 函数来描述例 4-7 中的 Rössler 方程，并利用该函数研究分别求解在该例中给定参数下和一组新参数 $a=0.3,b=2,c=3$ 下 Rössler 方程的数值解。

求解 选定附加参数为 a,b,c，根据上述的微分方程描述格式，可以编写出如下的 MATLAB 函数，来描述给出的常微分方程组。

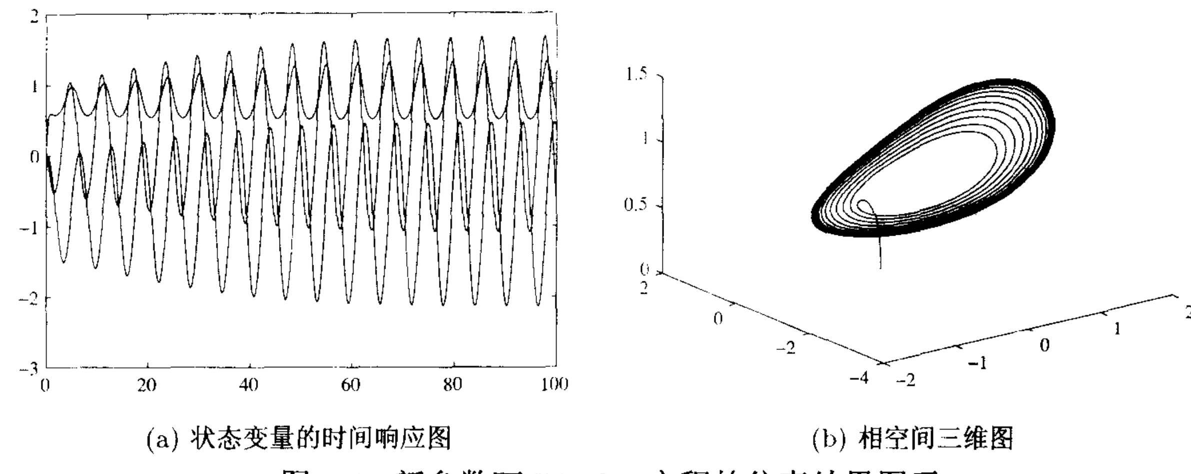
```
>> f2=@(t,x,a,b,c)[-x(2)-x(3); x(1)+a*x(2); b+(x(1)-c)*x(3)];
```

这样求解微分方程的语句可以类似地修改为

```
>> t_final=100; x0=[0;0;0]; a=0.3; b=2; c=3; opt=odeset;
   opt.RelTol=1e-8; [t,x]=ode45(f2,[0,t_final],x0,opt,a,b,c);
   plot(t,x), figure; plot3(x(:,1),x(:,2),x(:,3));
```

这时得出的二维和三维曲线，如图 4-4 (a)、图 4-4 (b) 所示。

4. 在控制理论中证明收敛的微分方程仿真研究



(a) 状态变量的时间响应图　　(b) 相空间三维图

图 4-4 新参数下 Rössler 方程的仿真结果图示

第 3 章给出了非线性系统的 Lyapunov 稳定性验证方法，这里将通过实际例子，利用仿真方法研究方程收敛的具体模式和收敛速度。

例 4-10 考虑例 3-57 中研究的微分方程模型 $\begin{cases} \dot{x}_1 = -x_1(x_1^2+x_2^2)+x_2 \\ \dot{x}_2 = -x_1 - x_2(x_1^2+x_2^2) \end{cases}$，在该例子中已经证明，该系统是渐近稳定的，试研究不同初值下的系统收敛速度。

求解 由给定的微分方程可以立即建立起 MATLAB 描述。随机选定 10 个初始状态，则可以绘制出系统的相平面曲线，如图 4-5 (a) 所示。可见，由各个初始状态出发的相平面曲线均将渐近地收敛到原点。

```
>> F=@(t,x)[-x(1)*(x(1)^2+x(2)^2)+x(2); -x(1)-x(2)*(x(1)^2+x(2)^2)];
   x10=floor(10*rand(10,1)); x20=floor(10*rand(10,1));
   for i=1:10
      [t,x]=ode45(F,[0,10],[x10(i); x20(i)]);
      plot(x(:,1),x(:,2)), hold on
   end
```

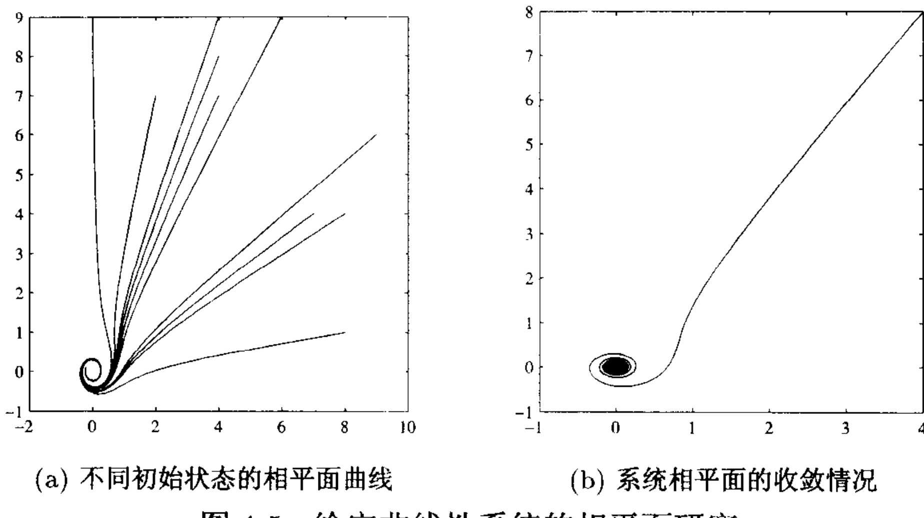
现在观察其中一条相平面曲线的收敛情况，设置终值时间为 1000 秒，则将得出如图 4-5 (b) 所示的相平面曲线，可见，虽然理论上相平面曲线渐近收敛到原点，但收敛的时间可能无限长。所以，系统的渐近稳定性并不是系统性能的惟一指标。

```
>> [t,x]=ode45(F,[0,1000],[x10(i); x20(i)]); plot(x(:,1),x(:,2))
```

4.2.4 微分方程转换

由前面介绍的微分方程求解函数和微分方程标准型可见，如果常微分方程由一个或多个高阶常微分方程给出，要得出该方程的数值解，则应该先将该方程变换成一阶显式常微分方程组。这里将分两种情况加以考虑。

1. 单个高阶常微分方程处理方法



(a) 不同初始状态的相平面曲线　　　(b) 系统相平面的收敛情况

图 4-5　给定非线性系统的相平面研究

假设一个高阶常微分方程的一般形式为

$$y^{(n)}=f(t,y,\dot{y},\cdots,y^{(n-1)}) \tag{4-2-9}$$

且已知输出变量 $y(t)$ 的各阶导数初始值为 $y(0),\dot{y}(0),\cdots,y^{(n-1)}(0)$，则可以选择一组状态变量 $x_1=y,x_2=\dot{y},\cdots,x_n=y^{(n-1)}$，这样，就可以将原高阶常微分方程模型变换成下面的一阶方程组形式

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=x_3\\ \quad\vdots\\ \dot{x}_n=f(t,x_1,x_2,\cdots,x_n)\end{cases} \tag{4-2-10}$$

且初值 $x_1(0)=y(0),x_2(0)=\dot{y}(0),\cdots,x_n(0)=y^{(n-1)}(0)$。这样，变换以后可以直接求取原方程的数值解了。

例 4-11 试用数值方法求出 Van der Pol 方程 $\ddot{y}+\mu(y^2-1)\dot{y}+y=0$ 的解，已知 $y(0)=-0.2,\dot{y}(0)=-0.7$。

求解 例 4-6 中已经演示过，该方程没有解析解，所以不能用解析方法求解该方程，只有依靠数值解法了。因为该方程不是一阶显式微分方程组，所以需要对该方程进行变换。选择状态变量 $x_1=y,x_2=\dot{y}$，则原方程可以变换成

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-\mu(x_1^2-1)x_2-x_1\end{cases}$$

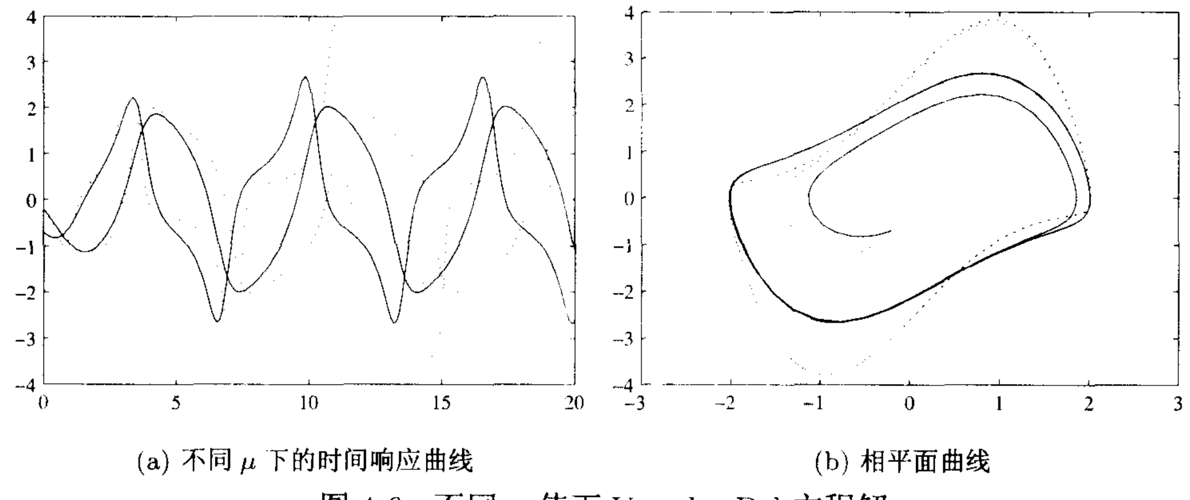
这里的 μ 是一个可变参数，如果对每一个要研究的 μ 值都编写一个函数则显得不方便，所以应该采用附加参数的概念将 μ 的值传给该函数，这样可以给出下面语句来描述此微分方程。

```
>> f=@(t,x,mu)[x(2);-mu*(x(1)^2-1)*x(2)-x(1)];
```

可见，在函数定义时多了一个 mu 项，该项应该在 ode45() 函数调用时传给 f() 函数。假定初值为 $x=[-0.2,-0.7]^{\mathrm{T}}$，则最终的求解函数格式为

```
>> x0=[-0.2; -0.7]; t_final=20;
   mu=1; [t1,y1]=ode45(f,[0,t_final],x0,[],mu);
   mu=2; [t2,y2]=ode45(f,[0,t_final],x0,[],mu);
   plot(t1,y1,t2,y2,':')
   figure; plot(y1(:,1),y1(:,2),y2(:,1),y2(:,2),':')
```

这样，在 $\mu=1,2$ 时的时间响应曲线和相平面曲线分别如图 4-6 (a)、图 4-6 (b) 所示。



(a) 不同 μ 下的时间响应曲线　　(b) 相平面曲线

图 4-6　不同 μ 值下 Van der Pol 方程解

注意，在定义函数时应该有个 flag 变量，其作用是用来指定初值的。即使初值不用指定，也必须有该变量占位。调用函数 ode45() 时也应该给出选项变量占位。在 ode45() 调用命令中的附加变量个数应该与匿名函数中的附加参数个数完全对应，否则将出现错误结果。

改变 μ 的值，令 $\mu=1000$，并设仿真终止时间为 3000，则可以采用下面的命令求解相应的 Van der Pol 方程。

```
>> x0=[2;0]; t_final=3000;
   mu=1000; [t,y]=ode45(f,[0,t_final],x0,[],mu);
```

经过长时间的等待，发现得出"Out of memory"的错误信息，表明求解失败。事实上，由于变步长所采用的步长过小，而要求的仿真终止时间比较大，导致输出的 y 矩阵过大，超出了计算机存储空间的容限。所以，这个问题不适合采用 ode45() 来求解，后面将采用刚性方程求解的算法来解决这个问题。

2. 高阶常微分方程组的变换方法

这里以两个高阶微分方程构成的微分方程组为例，介绍如何将之变换成一阶

显式常微分方程组。如果可以显式地将两个方程写成

$$\begin{cases} x^{(m)} = f(t, x, \dot{x}, \cdots, x^{(m-1)}, y, \cdots, y^{(n-1)}) \\ y^{(n)} = g(t, x, \dot{x}, \cdots, x^{(m-1)}, y, \cdots, y^{(n-1)}) \end{cases} \tag{4-2-11}$$

则仍旧可以选择状态变量 $x_1 = x, x_2 = \dot{x}, \cdots, x_m = x^{(m-1)}, x_{m+1} = y, x_{m+2} = \dot{y}, \cdots, x_{m+n} = y^{(n-1)}$，将原方程变换成

$$\begin{cases} \dot{x}_1 = x_2 \\ \quad\vdots \\ \dot{x}_m = f(t, x_1, x_2, \cdots, x_{m+n}) \\ \dot{x}_{m+1} = x_{m+2} \\ \quad\vdots \\ \dot{x}_{m+n} = g(t, x_1, x_2, \cdots, x_{m+n}) \end{cases} \tag{4-2-12}$$

再对初值进行相应的变换，就可以得出所期望的一阶微分方程组了。下面通过一个例子演示常微分方程组的转换与求解。

例 4-12 已知 Apollo 卫星的运动轨迹 (x, y) 满足下面的方程

$$\ddot{x} = 2\dot{y} + x - \frac{\mu^*(x+\mu)}{r_1^3} - \frac{\mu(x-\mu^*)}{r_2^3}, \quad \ddot{y} = -2\dot{x} + y - \frac{\mu^* y}{r_1^3} - \frac{\mu y}{r_2^3}$$

其中，$\mu = 1/82.45$, $\mu^* = 1 - \mu$, $r_1 = \sqrt{(x+\mu)^2 + y^2}$, $r_2 = \sqrt{(x-\mu^*)^2 + y^2}$，试在初值 $x(0) = 1.2$, $\dot{x}(0) = 0$, $y(0) = 0$, $\dot{y}(0) = -1.04935751$ 下进行求解，并绘制出 Apollo 位置的 (x, y) 轨迹。

求解 选择一组状态变量 $x_1 = x, x_2 = \dot{x}, x_3 = y, x_4 = \dot{y}$，这样就可以得出一阶常微分方程组为

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 2x_4 + x_1 - \mu^*(x_1 + \mu)/r_1^3 - \mu(x_1 - \mu^*)/r_2^3 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -2x_2 + x_3 - \mu^* x_3/r_1^3 - \mu x_3/r_2^3 \end{cases}$$

式中，$r_1 = \sqrt{(x_1+\mu)^2 + x_3^2}$, $r_2 = \sqrt{(x_1-\mu^*)^2 + x_3^2}$，且 $\mu = 1/82.45, \mu^* = 1 - \mu$。

有了数学模型描述，则可以立即写出其相应的 MATLAB 函数如下:

```
function dx=apolloeq(t,x)
mu=1/82.45; mu1=1-mu;
r1=sqrt((x(1)+mu)^2+x(3)^2); r2=sqrt((x(1)-mu1)^2+x(3)^2);
dx=[x(2); 2*x(4)+x(1)-mu1*(x(1)+mu)/r1^3-mu*(x(1)-mu1)/r2^3;
```

```
    x(4); -2*x(2)+x(3)-mu1*x(3)/r1^3-mu*x(3)/r2^3];
```

注意，这个方程由于存在中间运算与变量，所以不适合采用匿名函数或 inline() 函数进行描述，只能用 M 函数的形式描述微分方程。调用 ode45() 函数可以求出该方程的数值解为

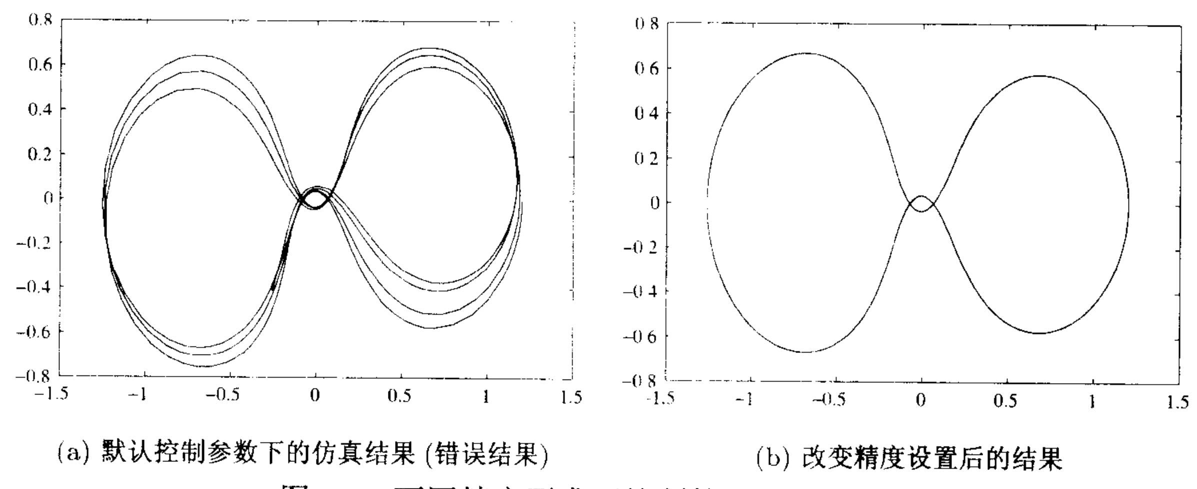
```
>> x0=[1.2; 0; 0; -1.04935751];
   tic, [t,y]=ode45(@apolloeq,[0,20],x0); toc
   length(t), plot(y(:,1),y(:,3))
```

得出的轨迹如图 4-7 (a) 所示。求解过程为 0.33 秒，共计算了 689 个点。

其实，这样直接得出的 Apollo 轨迹是不正确的，因为这时 ode45() 函数选择的默认精度控制 RelTol 设置得太大，从而导致较高的误差传递，最终导致错误的结果。如果想得出正确的结果，应该减小 RelTol 的值，例如将其减小到 10^{-6}，使用下面的语句进行仿真研究。

```
>> options=odeset; options.RelTol=1e-6;
   tic, [t1,y1]=ode45(@apolloeq,[0,20],x0,options); toc
   length(t1), plot(y1(:,1),y1(:,3)),
```

得出的轨迹如图 4-7 (b) 所示，这时所需求解时间 0.7 秒，计算点数 1873 个。可见，在新的默认精度下结果是完全不同的。这时，再进一步减小精度控制误差限也不会有太大的改进了。所以在仿真结束后有时有必要减小精度误差限 RelTol，看看得出的结果是否还相同，依此判定这样选择的精度要求是否合适，否则对求解结果是否正确没有把握。



(a) 默认控制参数下的仿真结果 (错误结果)　　(b) 改变精度设置后的结果

图 4-7　不同精度要求下绘制的 Apollo 轨迹图

用下面的 MATLAB 命令还能求出求解全程所采用的最小步长，并可以绘制出计算步长的曲线，如图 4-8 所示。

```
>> min(diff(t1)), plot(t1(1:end-1),diff(t1))
```

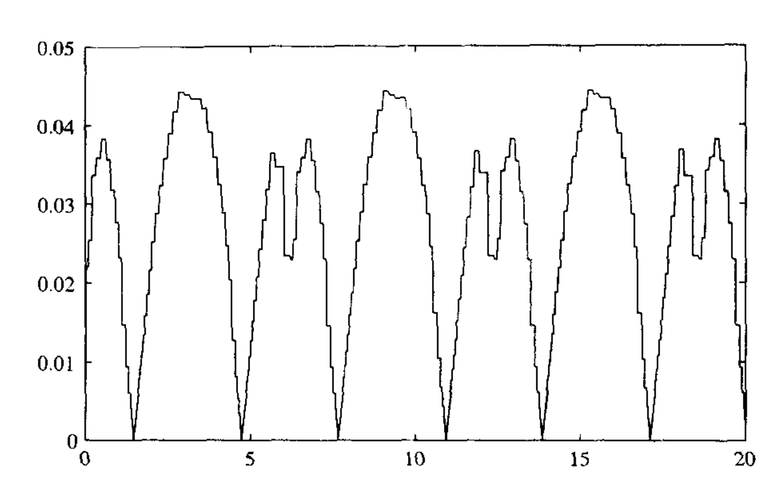


图 4-8　仿真过程中的计算步长

可见最小步长为 1.89×10^{-4}。从得出的图形可以看出变步长算法的意义，即在需要小步长时取小步长，而变换缓慢时取大步长计算，这样就可以保证以高效率求解方程。

由给出的计算步长图可以看出，部分时间段用了大于 0.04 的步长，这是一般定步长计算问题求解中不敢使用的大步长。为了保证某些具体时间点上的计算精度，还会自动采用 2×10^{-4} 的小步长。换言之，在这些点上如果采用比该值大的步长，则计算误差就不能保证在 10^{-6} 之下。考虑定步长计算的方式，如果想保证 10^{-6} 之下的误差，全程选择的步长就应该是这样的值，这样计算的点就要达到 10^5 个，是这里变步长算法的 56 倍。

例 4-13 试用定步长的四阶 Runge-Kutta 算法求解 Apollo 微分方程。

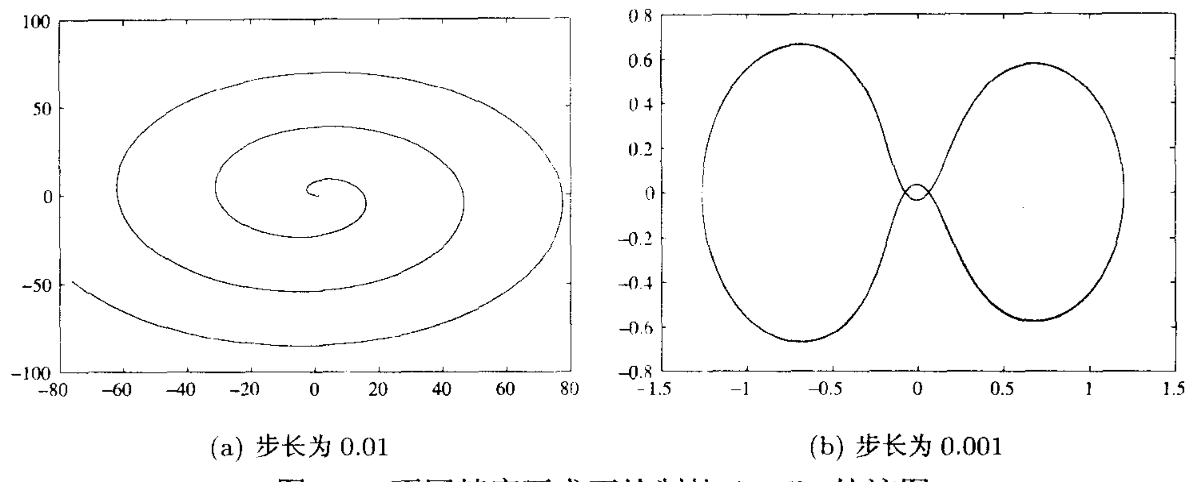
求解 用定步长的方法求解微分方程将面临两个问题：①如何选择步长；②如何确保求解的精度。前一个问题可以通过试凑的方法，从步长选择的角度看，选择很小的步长对保证求解精度有利，但计算量会明显增加。对前面介绍的 Apollo 轨迹方程，可以试着选择步长为 0.01，则用下面语句可以求解微分方程，并绘制出 Apollo 轨迹曲线，如图 4-9 (a) 所示。

```
>> x0=[1.2; 0; 0; -1.04935751];
   tic, [t1,y1]=rk_4(@apolloeq,[0,20,0.01],x0); toc
   plot(y1(:,1),y1(:,3))       % 绘制出轨迹曲线
```

所需的计算时间为 0.3 秒。显而易见，这样求解的结果是错误的，应该采用更小的步长求解，直至选择步长为 0.001，则可以求解微分方程，并绘制出更精确的轨迹曲线，如图 4-9 (b) 所示，但求解时间达到 13 秒，是变步长算法的 43 倍。

```
>> tic, [t2,y2]=rk_4(@apolloeq,[0,20,0.001],x0); toc
   plot(y2(:,1),y2(:,3))       % 绘制出轨迹曲线
```

其实，上面的结果在某些点上严格说来仍不能满足 10^{-6} 的误差限，只是形似变



(a) 步长为 0.01　　(b) 步长为 0.001

图 4-9　不同精度要求下绘制的 Apollo 轨迹图

步长得出的结果，所以在求解常微分方程组时建议采用变步长算法，而没有必要自己按照数值分析类教材中介绍的定步长算法去编写程序求解。

3. 隐式高阶微分方程组的变换方法

如果两个高阶微分方程都同时隐式地含有 $x^{(m)}$ 和 $y^{(n)}$ 项，则首先需要对之进行相应的处理，然后再用上述的方法进行最终变换。下面将通过一个例子加以说明。

例 4-14 假设系统模型以二元方程组形式给出 $\begin{cases} \ddot{x}+2\dot{y}x=2\ddot{y} \\ \ddot{x}\ddot{y}+3\dot{x}\ddot{y}+x\dot{y}-y=5 \end{cases}$，试将其转换成一阶微分方程组。

求解 可见，这两个方程均同时含有 $\ddot{x}$ 和 $\ddot{y}$，所以仍可以选择一组状态变量 $x_1=x, x_2=\dot{x}, x_3=y, x_4=\dot{y}$，我们的目的是先消去其中一个高阶导数，求解第一个式子，得出 $\ddot{y}$ 的解析表达式为 $\ddot{y}=\dot{y}x+\dfrac{\ddot{x}}{2}$，然后将它代入第二个式子中，则可以解出 $\ddot{x}$ 为 $\ddot{x}=\dfrac{2y+10-2x\dot{y}-6x\dot{x}\dot{y}}{2\dot{y}+3\dot{x}}$，这样可以写出其状态方程表示为

$$\dot{x}_2=\frac{2x_3+10-2x_1x_4-6x_1x_2x_4}{2x_4+3x_2},\quad \dot{x}_4=\frac{x_3+5-x_1x_4+2x_1x_4^2}{2x_4+3x_2}$$

综上所述，可以列写出方程的一阶微分方程组表示为

$$\begin{cases} \dot{x}_1=x_2 \\ \dot{x}_2=\dfrac{2x_3+10-2x_1x_4-6x_1x_2x_4}{2x_4+3x_2} \\ \dot{x}_3=x_4 \\ \dot{x}_4=\dfrac{x_3+5-x_1x_4+2x_1x_4^2}{2x_4+3x_2} \end{cases}$$

对于更复杂的问题来说，手工变换的难度将很大，所以如果可能，可以采用

计算机去求解有关方程，获得解析解。如果不能获得方程的解析解，也需要在描写一阶常微分方程组时列写出式子，得出问题的数值解。

4.2.5 矩阵微分方程的变换与求解方法

前面介绍的微分方程数值解均是对向量型状态变量 $\boldsymbol{x}(t)$ 进行求解的，所以需要事先将该方程化成向量型方程，然后调用求解函数，如 `ode45()` 直接求解该方程。在实际应用中经常遇到其他形式的微分方程模型，如在机器人等学科中的 Lagrange 方程，对应的微分方程为下面的矩阵微分方程

$$\boldsymbol{M}\ddot{\boldsymbol{X}}+\boldsymbol{C}\dot{\boldsymbol{X}}+\boldsymbol{K}\boldsymbol{X}=\boldsymbol{F}u(t) \tag{4-2-13}$$

其中 $\boldsymbol{M},\boldsymbol{C},\boldsymbol{K}$ 为 $n\times n$ 矩阵，$\boldsymbol{X},\boldsymbol{F}$ 均为 $n\times 1$ 列向量。引入 $\boldsymbol{x}_1=\boldsymbol{X}$，$\boldsymbol{x}_2=\dot{\boldsymbol{X}}$，则 $\dot{\boldsymbol{x}}_1=\boldsymbol{x}_2$，$\dot{\boldsymbol{x}}_2=\ddot{\boldsymbol{X}}$。由式 (4-2-13) 可见，$\ddot{\boldsymbol{X}}=\boldsymbol{M}^{-1}\Big[\boldsymbol{F}u(t)-\boldsymbol{C}\dot{\boldsymbol{X}}-\boldsymbol{K}\boldsymbol{X}\Big]$。这样选择状态变量 $\boldsymbol{x}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}}]^{\mathrm{T}}$，则可以写出系统的状态方程模型

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}\boldsymbol{x}_2(t)\\ \boldsymbol{M}^{-1}\Big[\boldsymbol{F}u(t)-\boldsymbol{C}\boldsymbol{x}_2(t)-\boldsymbol{K}\boldsymbol{x}_1(t)\Big]\end{bmatrix} \tag{4-2-14}$$

可见，该微分方程是关于 $\boldsymbol{x}(t)$ 列向量的显式一阶微分方程组，所以可以通过 MATLAB 提供的函数直接求解。下面将通过例子给出问题的求解方法。

另外，如果各个矩阵 $\boldsymbol{M}$，$\boldsymbol{C}$，$\boldsymbol{K}$ 与 $\boldsymbol{F}$ 均和 $\boldsymbol{X}$ 无关，则该微分方程为线性微分方程，还可以通过简单的变换将其改写为相应的线性状态方程模型为

$$\begin{bmatrix}\dot{\boldsymbol{x}}_1(t)\\ \dot{\boldsymbol{x}}_2(t)\end{bmatrix}=\begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1(t)\\ \boldsymbol{x}_2(t)\end{bmatrix}+\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{M}^{-1}\boldsymbol{F}\end{bmatrix}u(t) \tag{4-2-15}$$

例 4-15 已知二级倒立摆系统的数学模型由下式给出[2]

$$\boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}+\boldsymbol{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}=\boldsymbol{F}(\boldsymbol{\theta})$$

其中 $\boldsymbol{\theta}=[a,\theta_1,\theta_2]^{\mathrm{T}}$，且 a 为小车位置，θ_1,θ_2 分别为下摆杆、上摆杆与垂直方向夹角，倒立摆系统的各个矩阵为

$$\boldsymbol{M}(\boldsymbol{\theta})=\begin{bmatrix}m_{\mathrm{c}}+m_1+m_2 & (0.5m_1+m_2)L_1\cos\theta_1 & 0.5m_2L_2\cos\theta_2\\ (0.5m_1+m_2)L_1\cos\theta_1 & (m_1/3+m_2)L_1^2 & 0.5m_2L_1L_2\cos\theta_1\\ 0.5m_2L_2\cos\theta_2 & 0.5m_2L_1L_2\cos\theta_1 & m_2L_2^2/3\end{bmatrix}$$

$$\boldsymbol{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})=\begin{bmatrix}0 & -(0.5m_1+m_2)L_1\dot{\theta}_1\sin\theta_1 & -0.5m_2L_2\dot{\theta}_2\sin\theta_2\\ 0 & 0 & 0.5m_2L_1L_2\dot{\theta}_2\sin(\theta_1-\theta_2)\\ 0 & -0.5m_2L_1L_2\dot{\theta}_1\sin(\theta_1-\theta_2) & 0\end{bmatrix}$$

$$F(\theta)=\begin{bmatrix} u(t) \\ (0.5m_1+m_2)L_1 g\sin\theta_1 \\ 0.5m_2L_2 g\sin\theta_2 \end{bmatrix}$$

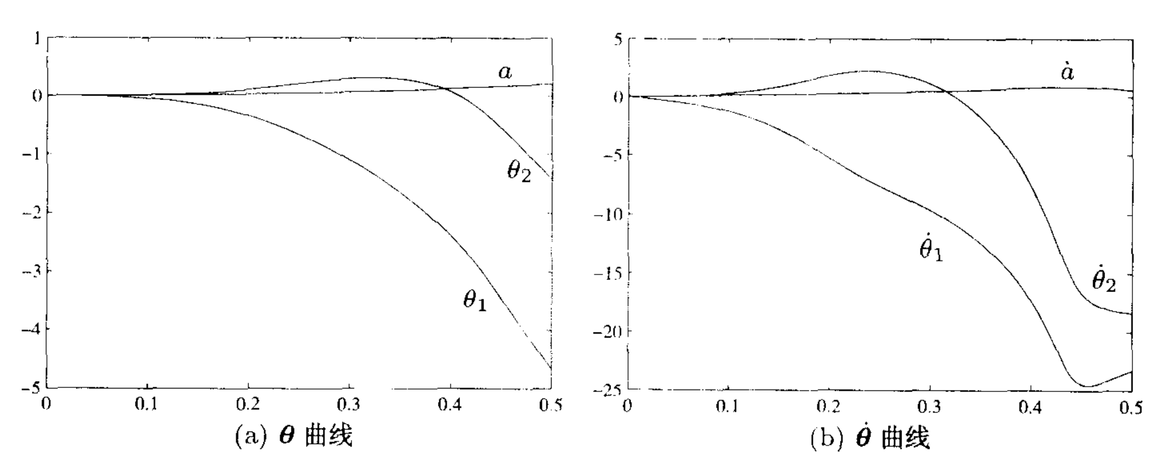
已知二级倒立摆的参数为 $m_c=0.85\text{kg}$, $m_1=0.04\text{kg}$, $m_2=0.14\text{kg}$, $L_1=0.1524\text{m}$, $L_2=0.4318\text{m}$，试用数值方法求解系统的阶跃响应。

求解 可见，因为系数矩阵 $\boldsymbol{M}(\theta_1,\theta_2)$, $\boldsymbol{C}(\theta_1,\theta_2)$ 和 $\boldsymbol{F}(\theta_1,\theta_2)$ 都含有状态变量 $\boldsymbol{x}$ 的非线性项，如 θ_1 的正弦余弦项等，所以原来的系统不是线性微分方程，不能直接写成式 (4-2-14) 的形式。引入附加变量 $\boldsymbol{x}_1=\boldsymbol{\theta}$, $\boldsymbol{x}_2=\dot{\boldsymbol{\theta}}$，并构造状态变量 $\boldsymbol{x}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}}]^{\mathrm{T}}$，则可以用下面的语句描述上述的一阶显式微分方程模型

```
function dx=inv_pendulum(t,x,u,mc,m1,m2,L1,L2,g)
M=[mc+m1+m2, (0.5*m1+m2)*L1*cos(x(2)), 0.5*m2*L2*cos(x(3))
   (0.5*m1+m2)*L1*cos(x(2)),(m1/3+m2)*L1^2,0.5*m2*L1*L2*cos(x(2))
   0.5*m2*L2*cos(x(3)),0.5*m2*L1*L2*cos(x(2)),m2*L2^2/3];
C=[0,-(0.5*m1+m2)*L1*cos(x(5))*sin(x(2)),-0.5*m2*L2*x(6)*sin(x(3))
   0, 0, 0.5*m2*L1*L2*x(6)*sin(x(2)-x(3))
   0, -0.5*m2*L1*L2*x(5)*sin(x(2)-x(3)), 0];
F=[u; (0.5*m1+m2)*L1*g*sin(x(2)); 0.5*m2*L2*g*sin(x(3))];
dx=[x(4:6); inv(M)*(F-C*x(4:6))];
```

若用阶跃信号激励该系统，则可以由下面的语句直接求出方程的数值解，如图 4-10 所示。

```
>> opt=odeset; opt.RelTol=1e-8; u=1; mc=0.85;
   m1=0.04; m2=0.14; L1=0.1524; L2=0.4318; g=9.81;
   [t,x]=ode45(@inv_pendulum,[0,0.5],zeros(6,1),opt,...
              u,mc,m1,m2,L1,L2,g);
   plot(t,x(:,1:3)), figure; plot(t,x(:,4:6))
```



(a) $\boldsymbol{\theta}$ 曲线 (b) $\dot{\boldsymbol{\theta}}$ 曲线

图 4-10 二级倒立摆的阶跃响应曲线

值得指出的是，由于二级倒立摆系统是自然不稳定系统，所以若给系统施加阶跃

输入是没有任何意义的，应该给其施加某些控制才能使其到达稳定状态。

4.2.6 微分方程数值解正确性的验证

前面通过例子曾演示过，若仿真控制参数选择不当，如相对误差限，则可能得出不可信的结果，甚至是错误的结果。所以在方程求解结束之后，应该对仿真结果进行检验。然而所需求解的问题又不存在解析解，怎么检验所得结果的正确性呢？一种可行的方法是修改仿真控制参数，如可以接受的误差限，例如将 RelTol 选项设置成一个更小的值，观察所得的结果，看看是不是和上次得出的结果完全一致，如果存在不能接受的差异，则应该考虑再进一步减小误差限。另外，同样的问题选择不同的微分方程求解算法也可以检验所得结果的正确性。

4.3 特殊微分方程的数值解

从 4.2 节的介绍和例子看，一般常微分方程组均可以转换成一阶显式常微分方程组，然后通过给定的算法及 MATLAB 求解函数，如 ode45() 直接求出方程的数值解。从前面的例子还可以看出，ode45() 函数有时失效，所以应该引入一类方程，即刚性微分方程的专门求解函数来解决这样的问题。另外，微分代数方程、隐式微分方程、延迟微分方程和切换微分方程等也是需要引入的特殊的微分方程类型，本节将着重介绍这些方程的求解问题。

4.3.1 刚性微分方程的求解

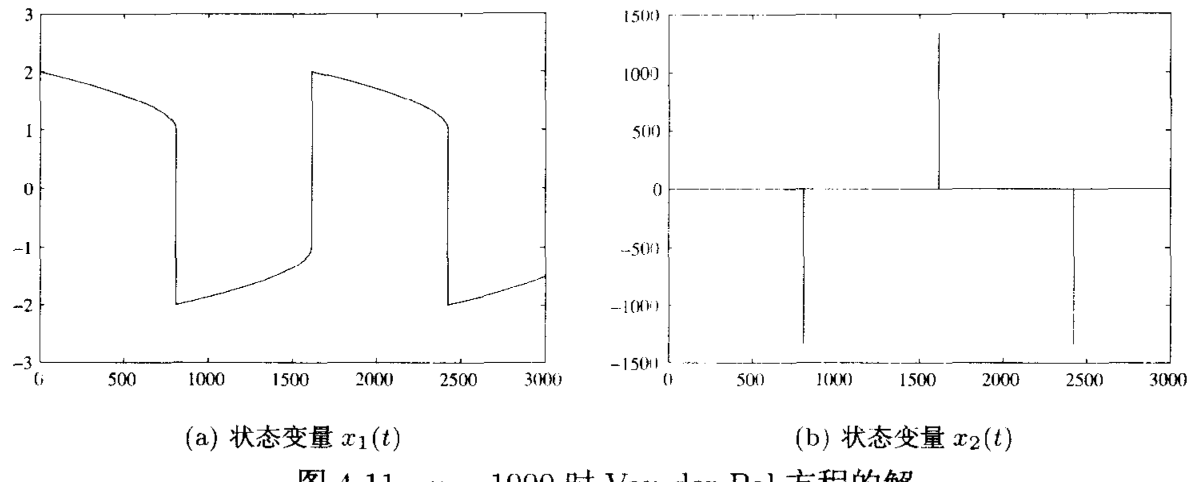
在许多领域中，经常遇到一类特殊的常微分方程，其中一些解变化缓慢，另一些变化快，且相差较悬殊，这类方程常常称为刚性方程，又称为 Stiff 方程。刚性问题一般不适合由 ode45() 这类函数求解，而应该采用 MATLAB 求解函数 ode15s()，该函数的调用格式和 ode45() 完全一致。

例 4-16 试求解 $\mu=1000$ 时的 Van der Pol 方程的数值解。

求解 仿照前面的例子可以给出如下的 MATLAB 命令：

```
>> f=@(t,x,mu)[x(2);-mu*(x(1)^2-1)*x(2)-x(1)];
   h_opt=odeset; h_opt.RelTol=1e-6; x0=[2;0]; t_final=3000;
   tic, mu=1000; [t,y]=ode15s(f,[0,t_final],x0,h_opt,mu); toc
   plot(t,y(:,1)); figure; plot(t,y(:,2))
```

所需仿真时间为 1.943 秒。可见，用刚性方程求解函数可以快速求出该方程的数值解，并将两个状态变量的时间曲线分别绘制出来，如图 4-11 所示。从得出的图形可以看出，$x_1(t)$ 曲线变化较平滑，而 $x_2(t)$ 变化在某些点上较快，所以当 $\mu=1000$ 时，Van der Pol 方程属于典型的刚性方程，应该采用刚性方程的函数求解。



(a) 状态变量 $x_1(t)$ (b) 状态变量 $x_2(t)$

图 4-11 $\mu = 1000$ 时 Van der Pol 方程的解

例 4-17 在传统的有关常微分方程数值解的教科书[3]中，都认为下面的微分方程是刚性的，试用 MATLAB 语言求解该微分方程。

$$\dot{\boldsymbol{y}} = \begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix} \boldsymbol{y},\ \boldsymbol{y}_0 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

求解 该方程的解析解可以通过 MATLAB 符号运算工具箱中语句直接求出。

```
>> syms t; A=[-21,19,-20; 19,-21,20; 40,-40,-40];
   y0=[1; 0; -1]; y=expm(A*t)*y0
```

原方程的解析解为 $\boldsymbol{y}(t) = \begin{bmatrix} 0.5\mathrm{e}^{-2t} + 0.5\mathrm{e}^{-40t}(\cos 40t + \sin 40t) \\ 0.5\mathrm{e}^{-2t} - 0.5\mathrm{e}^{-40t}(\cos 40t + \sin 40t) \\ \mathrm{e}^{-40t}(\sin 40t - \cos 40t) \end{bmatrix}$。

现在考虑该问题的数值求解方法。根据原始问题，可以立即写出描述原微分方程的 MATLAB 表示为

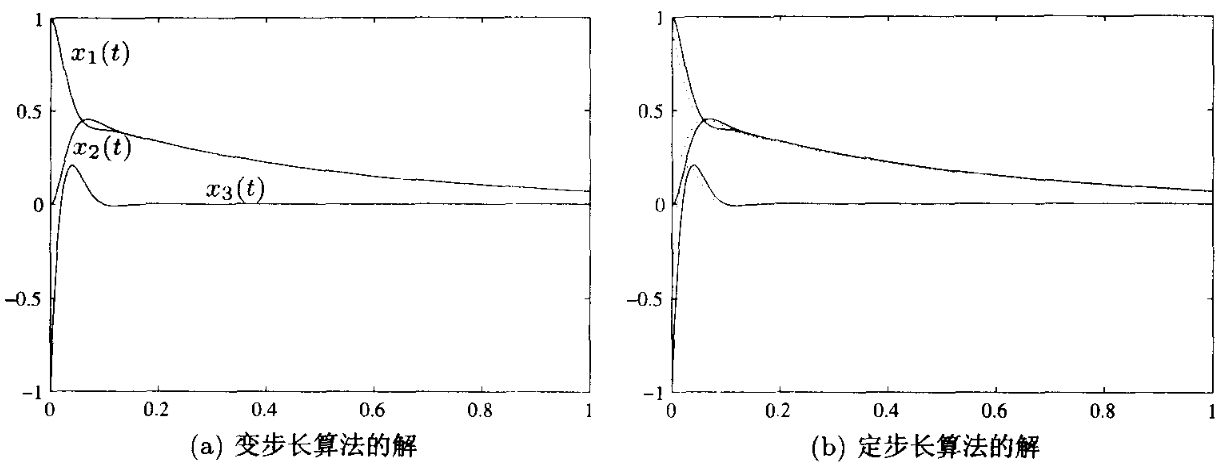
```
>> f=@(t,x)[-21,19,-20; 19,-21,20; 40,-40,-40]*x;
```

利用下面的 MATLAB 语句，可以得出方程的数值解为

```
>> opt=odeset; opt.RelTol=1e-6; [t1,y]=ode45(f,[0,1],[1;0;-1],opt);
   y1=subs(y,'t',t1); plot(t1,y,t1,y1,':')
```

原方程的解析解和数值解如图 4-12 (a) 所示。可以看出，问题的数值解的精度还是比较高的，计算速度相对也较快，但从这里似乎看不出原问题的刚性所在。究其原因，是因为在 MATLAB 下采用了变步长的算法，它可以依照要求的精度自动地修正步长，所以感受不到它是个刚性问题。

如果用户想采用定步长方法，利用前面编写的四阶 Runge-Kutta 定步长算法程序 rk_4()，再用下面的语句就能求解原方程了。



(a) 变步长算法的解　　(b) 定步长算法的解

图 4-12　给定的传统刚性问题解法比较

```
>> tic, [t2,y2]=rk_4(f,[0,1,0.01],[1;0;-1]); toc
   plot(t1,y1,t2,y2,':')
```

这时的求解时间为 0.21 秒。这样得出的曲线与解析解的对比见图 4-12 (b)。从计算的结果看，显然采用定步长的算法在取较大的步长时，得出的解是不正确的。减小步长时，直至减小到 0.0001 时，仍能从得出的结果看出与解析解的差距，而这时的计算时间为 26 秒，是前面 ode45() 变步长算法所需时间的 162 倍。所以在实际应用中最好采用变步长算法。

从得出的曲线可以看出，相比之下，这 3 条曲线的变化速度差别不是特别悬殊，在以前计算能力受限时被误认为是刚性问题，而在 MATLAB 下则可以由普通的函数求解。

由此可以得出结论，许多传统的刚性问题采用 MATLAB 的普通求解函数就可以直接解出，而不必刻意地去选择刚性问题的解法。当然，在有些问题的求解中确实需要采用刚性问题的解法，这将在例 4-18 中加以说明。

例 4-18 考虑下面的常微分方程

$$\begin{cases} \dot{y}_1 = 0.04(1-y_1) - (1-y_2)y_1 + 0.0001(1-y_2)^2 \\ \dot{y}_2 = -10^4 y_1 + 3000(1-y_2)^2 \end{cases}$$

其中，该方程的初值为 $y_1(0) = 0, y_2(0) = 1$。取计算区间为 $t \in (0, 100)$，试选择合适的算法得出该方程的数值解。

求解 根据给出的微分方程，可以由下面的语句直接求其数值解

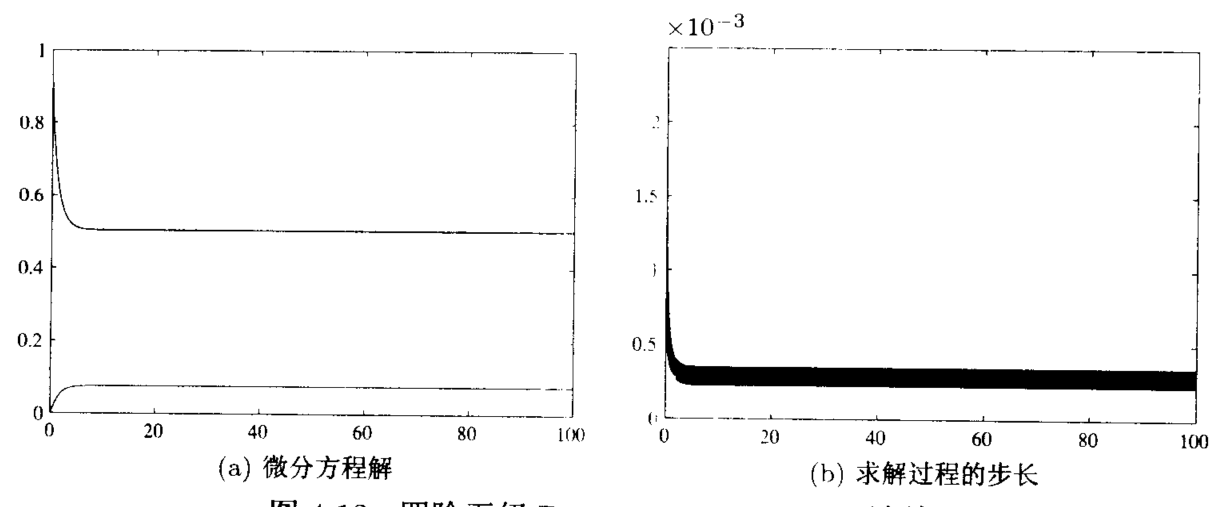
```
>> f=@(t,y)[0.04*(1-y(1))-(1-y(2))*y(1)+0.0001*(1-y(2))^2;...
          -1e4*y(1)+3000*(1-y(2))^2];
   tic,[t2,y2]=ode45(f,[0,100],[0;1]); toc
```

```
length(t2), plot(t2,y2)
```

经过长时间等待，可以得出原问题的数值解，如图 4-13 (a) 所示。可以看出，调用普通的解法函数 ode45() 计算所需的时间过长，对这个例子来说，计算的点高达 35 万。再分析变步长解法所使用的步长，语句如下:

```
>> format long, [min(diff(t2)), max(diff(t2))]
   plot(t2(1:end-1),diff(t2))
```

则可以看出，由于设定的精度要求较高，不得不采用小步长来解决问题。实际的步长如图 4-13 (b) 所示。可见在大部分时间内，所采用的步长小于 0.0004，这使得解题时间大大增加。



(a) 微分方程解　　(b) 求解过程的步长

图 4-13　四阶五级 Runge-Kutta-Felhberg 法结果

考虑用 ode15s() 替代 ode45()，可以得出如下的结果:

```
>> opt=odeset; opt.RelTol=1e-6;
   tic,[t1,y1]=ode15s(f,[0,100],[0;1],opt); toc
   length(t1), plot(t1,y1)
```

这时所需时间为 0.17 秒，计算点数为 169 个。可见，解题时间大大减小，效率提高了 840 倍，得出的曲线则几乎完全一致。

另外，值得指出的是，虽然用 MATLAB 的 M-函数、inline() 函数和匿名函数均可以描述微分方程模型，但求解计算量大的问题时所用的时间和求解的效率是大不相同的。对本例来说，匿名函数需要 56 秒，M-函数需要 143 秒，而 inline() 描述所需的求解时间高达 20 分钟! 所以在求解刚性方程时，可以优先考虑采用匿名函数。

4.3.2 隐式微分方程求解

所谓隐式微分方程就是那些不能转换成式 (4-2-1) 中一阶显式常微分方程组

的微分方程。MATLAB 语言早期版本中未提供能直接求解隐式微分方程的算法及函数，所以仍可以借用显式微分方程求解的函数来求解这类问题。本节将通过两个例子来演示隐式微分方程的求解方法与应用。

例 4-19 给定的隐式微分方程为

$$\begin{cases} \sin x_1\dot{x}_1 + \cos x_2\dot{x}_2 + x_1 = 1 \\ -\cos x_2\dot{x}_1 + \sin x_1\dot{x}_2 + x_2 = 0 \end{cases}$$

已知 $x_1(0) = x_2(0) = 0$，试求出该方程的数值解。

求解 令 $\boldsymbol{x} = [x_1, x_2]^{\mathrm{T}}$，则可以将原方程改写成矩阵形式 $\boldsymbol{A}(\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{B}(\boldsymbol{x})$，其中，

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} \sin x_1 & \cos x_2 \\ -\cos x_2 & \sin x_1 \end{bmatrix}, \quad \boldsymbol{B}(\boldsymbol{x}) = \begin{bmatrix} 1 - x_1 \\ -x_2 \end{bmatrix}$$

如果能证明 $\boldsymbol{A}(\boldsymbol{x})$ 为非奇异的矩阵，则直接能将该方程变换成标准的显式一阶微分方程组的形式，即 $\dot{\boldsymbol{x}} = \boldsymbol{A}^{-1}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})$，套用各种 MATLAB 函数即可求解对应的方程。事实上，由于无法严格证明 $\boldsymbol{A}(\boldsymbol{x})$ 矩阵是非奇异矩阵，故可以大胆地假设该矩阵为非奇异矩阵，利用 MATLAB 语言试解该方程，如果在求解过程中不出现 $\boldsymbol{A}(\boldsymbol{x})$ 为奇异矩阵的警告信息，则说明对求出的解不存在 $\boldsymbol{A}(\boldsymbol{x})$ 为奇异矩阵的现象，故得出的解是有效的。如果求解过程中确实出现矩阵奇异的信息，则得出的解没有实际意义。

对于这里要研究的隐式微分方程模型，可以用匿名函数描述微分方程，并可以给出下面的命令求解方程:

```
>> f=@(t,x)inv([sin(x(1)) cos(x(2)); -cos(x(2)) sin(x(1))])...
       *[1-x(1); -x(2)];  opt=odeset; opt.RelTol=1e-6;
   [t,x]=ode45(f,[0,10],[0; 0],opt); plot(t,x)
```

同时将绘制出状态变量的时间曲线，如图 4-14 所示。在整个求解的过程中没有得出有关矩阵奇异的错误信息，故得出的结果是可信的。

例 4-20 前面的例子很简单，可以将其立即变换成显式微分方程。现在考虑一个更复杂的隐式微分方程组

$$\begin{cases} \ddot{x}\sin\dot{y} + \ddot{y}^2 = -2\mathrm{e}^{-\dot{x}}xy + x\ddot{x}\dot{y} \\ x\ddot{x}\ddot{y} + \cos\ddot{y} = 3y\dot{x}\mathrm{e}^{-x} \end{cases}$$

假设该方程具有初始状态为 $\boldsymbol{x} = [1, 0, 0, 1]^{\mathrm{T}}$，试求取该微分方程的数值解。

求解 从给出的方程可见，不能像前面的方法变换成简单的微分方程。下面将研究这类隐式微分方程的求解方法。

MATLAB 7.0 版本的新函数 `ode15i()` 可以直接用于隐式微分方程的求解。若隐式微分方程的数学描述为

$$\boldsymbol{F}[t, \boldsymbol{x}(t), \dot{\boldsymbol{x}}(t)] = \boldsymbol{0}, \quad \text{且 } \boldsymbol{x}(0) = \boldsymbol{x}_0,\ \dot{\boldsymbol{x}}(0) = \dot{\boldsymbol{x}}_0 \tag{4-3-1}$$

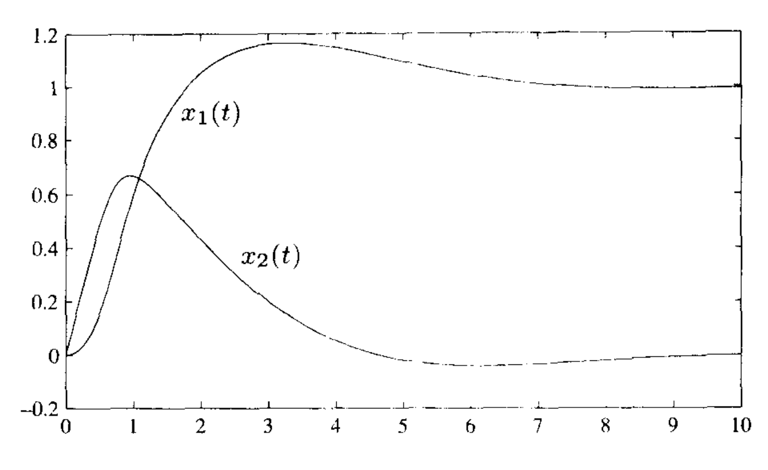


图 4-14 隐式方程的时间响应曲线

则可以编写一个函数 fun() 描述该隐式微分方程，然后用 decic() 函数求出未完全定义的初值条件，再调用 ode15i() 函数即可以求解该隐式微分方程如下：

$$[\boldsymbol{x}_0^*,\dot{\boldsymbol{x}}_0^*]=\texttt{decic}(\texttt{fun},t_0,\boldsymbol{x}_0,\boldsymbol{x}_0^{\rm F},\dot{\boldsymbol{x}}_0,\dot{\boldsymbol{x}}_0^{\rm F})$$
$$\texttt{res=ode15i}(\texttt{fun},\texttt{tspan},\boldsymbol{x}_0^*,\dot{\boldsymbol{x}}_0^*)$$

隐式微分方程不同于一般显示微分方程，求解之前需要给出 $(\boldsymbol{x}_0,\dot{\boldsymbol{x}}_0)$，它们不能任意赋值，只能有 n 个是独立的，其余的需要用隐式方程求解，否则将可能出现矛盾的初始条件。所以在实际求解过程中，如果不能确定 $\dot{\boldsymbol{x}}_0^*$ 值，则应该先调用 decic() 函数得出相容的初值。在函数调用中 $(\boldsymbol{x}_0,\dot{\boldsymbol{x}}_0)$ 为任意给定的初值，$\boldsymbol{x}_0^{\rm F}$ 和 $\dot{\boldsymbol{x}}_0^{\rm F}$ 均为 n 维列向量，其值为 1 表示需要保留的初值，为 0 表示需要求解的初值项，通过方程求解将得出相容的初值 $\boldsymbol{x}_0^*,\dot{\boldsymbol{x}}_0^*$，该初值可以直接用于隐式微分方程求解函数 ode15i()，该函数的返回变量 res.x 和 res.y 将分别为 $\boldsymbol{t}$ 和 $\boldsymbol{x}$。下面将通过具体例子演示隐式微分方程的求解方法。

例 4-21 试用隐式微分方程求解的方法解出例 4-20 中给出的隐式微分方程。

求解 选择状态变量 $x_1=x, x_2=\dot{x}, x_3=y, x_4=\dot{y}$，则原方程可以变换成

$$\begin{cases} \dot{x}_1-x_2=0 \\ \dot{x}_2\sin x_4+\dot{x}_4^2+2\mathrm{e}^{-x_2}x_1x_3-x_1\dot{x}_2x_4=0 \\ \dot{x}_3-x_4=0 \\ x_1\dot{x}_2\dot{x}_4+\cos\dot{x}_4-3\mathrm{e}^{-x_1}x_3x_2=0 \end{cases}$$

这样，可以编写出如下所示的描述隐式微分方程的匿名函数，调用下面的语句即可以求解隐式微分方程，且可绘制出该方程解的时间曲线，如图 4-15 所示。

```
>> f=@(t,x,xd)[xd(1)-x(2);
     xd(2)*sin(x(4))+xd(4)^2+2*exp(-x(2))*x(1)*x(3)-x(1)*xd(2)*x(4);
     xd(3)-x(4);
```

```
  x(1)*xd(2)*xd(4)+cos(xd(4))-3*exp(-x(1))*x(3)*x(2)];
x0=[1,0,0,1]; xd0=[0;1;1;-1]; x0F=[1 1 1 1]'; xd0F=[];
[x0,xd0]=decic(f,0,x0,x0F,xd0,xd0F)
res=ode15i(f,[0,2],x0,xd0);  % 求解隐式微分方程
plot(res.x,res.y)  % 绘制各个状态的时间响应曲线
```

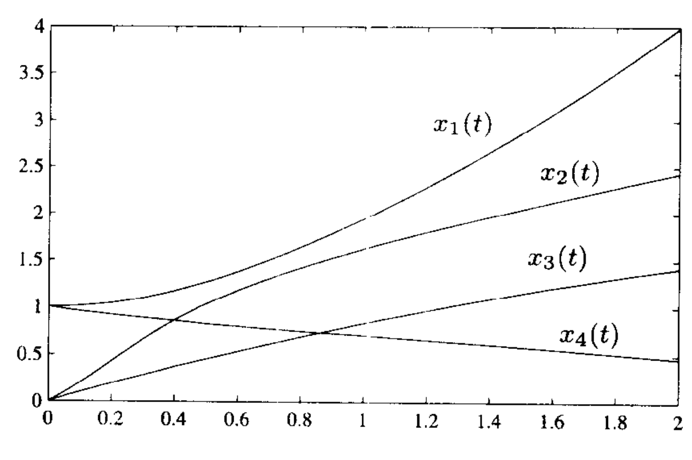


图 4-15 隐式方程的时间响应曲线

在初值处理上，因为已知 $\boldsymbol{x}_0$，所以其中 $\boldsymbol{x}_0^{\mathrm{F}}$ 应该设置成幺向量，$\dot{\boldsymbol{x}}_0^{\mathrm{F}}$ 可以设置成零向量或空向量，表示 $\dot{\boldsymbol{x}}_0$ 需要由 decic() 函数求出。由 decic() 函数可以求出相容的导数初值为 $\dot{\boldsymbol{x}}_0 = [0, 1.6833, 0, -0.5166]^{\mathrm{T}}$。

4.3.3 微分代数方程与广义系统的求解

在前面的介绍中，所介绍的常微分方程数值解法主要是针对能够转换成一阶常微分方程组的类型，假设其中的一些微分方程退化为代数方程，则用前面介绍的算法无法求解，必须借助微分代数方程的特殊解法。

所谓微分代数方程 (differential algebraic equation，DAE)，是指在微分方程中，某些变量间满足代数方程的约束，所以这样的方程不能用前面介绍的常微分方程解法直接进行求解。假设微分方程的更一般形式可以写成

$$\boldsymbol{M}(t,\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{f}(t,\boldsymbol{x}) \tag{4-3-2}$$

描述 $\boldsymbol{f}(t,\boldsymbol{x})$ 的方法和普通常微分方程完全一致，而对真正的微分代数方程来说，$\boldsymbol{M}(t,\boldsymbol{x})$ 矩阵为奇异矩阵，在微分代数方程求解程序中应该由求解选项中的成员变量 Mass 来表示该矩阵，考虑了这些因素则可以立即求解方程的解了。注意，微分代数方程的求解函数建议使用 ode15s() 等刚性方程求解函数，因为 ode45() 函数有时会出现不收敛现象。

如果 $\boldsymbol{M}(t,\boldsymbol{x})$ 为奇异矩阵，则式 (4-3-2) 也可以表示一种特殊的状态方程模型，这类微分方程的某些方程已经退化成代数方程。这样状态方程模型对应的系统又称为广义系统[4]，或奇异系统。

例 4-22 考虑下面给出的微分代数方程

$$\begin{cases} \dot{x}_1 = -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ \dot{x}_2 = 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ x_1 + x_2 + x_3 - 1 = 0 \end{cases}$$

并已知初始条件为 $x_1(0) = 0.8$, $x_2(0) = x_3(0) = 0.1$，试求取该方程的数值解。

求解 可以看出，最后的一个方程为代数方程，可以视之为 3 个状态变量间的约束关系。用矩阵的形式可以表示该微分代数方程为

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ x_1 + x_2 + x_3 - 1 \end{bmatrix}$$

这样就可以写出相应的匿名函数如下:

```
>> f=@(t,x)[-0.2*x(1)+x(2)*x(3)+0.3*x(1)*x(2);,...
       2*x(1)*x(2)-5*x(2)*x(3)-2*x(2)*x(2); x(1)+x(2)+x(3)-1];
```

可以将 M 矩阵输入给 MATLAB 工作空间，并赋给求解控制参数的 Mass 属性，则在命令窗口中可以给出如下命令:

```
>> M=[1,0,0; 0,1,0; 0,0,0]; options=odeset; options.Mass=M;
   x0=[0.8; 0.1; 0.1]; [t,x]=ode15s(f,[0,20],x0,options);
   plot(t,x)
```

由上面的语句可以得出此微分代数方程的解，如图 4-16 所示。

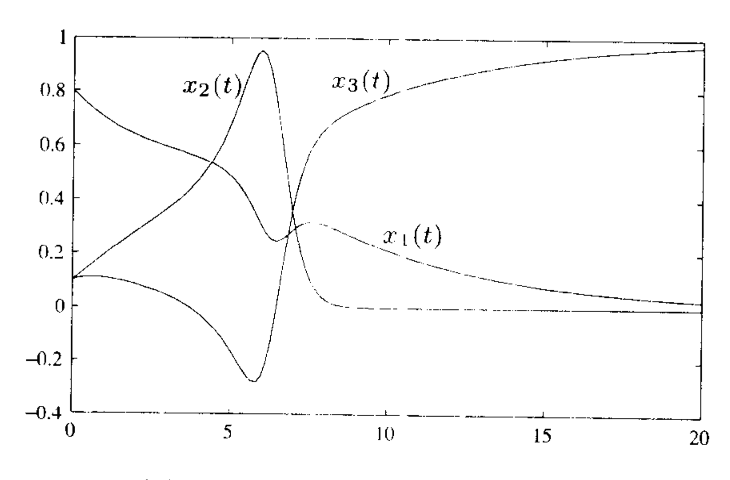


图 4-16 微分代数方程的数值解

事实上，有些微分代数方程可以转换成常微分方程求解。例如，在本例中，可以

从约束式子中求出 $x_3(t)=1-x_1(t)-x_2(t)$，将其代入其他两个微分方程式子，则有

$$\begin{cases}\dot{x}_1=-0.2x_1+x_2[1-x_1(t)-x_2(t)]+0.3x_1x_2\\ \dot{x}_2=2x_1x_2-5x_2[1-x_1(t)-x_2(t)]-2x_2^2\end{cases}$$

根据该方程可以用匿名函数描述上述的微分方程，这样则可以用下面的命令求解变换后的微分方程组，从而最终得出原微分代数方程的解，所得出的解与前面的直接解法得出的完全一致。

```
>> x0=[0.8; 0.1];
   fDae=@(t,x)[-0.2*x(1)+x(2)*(1-x(1)-x(2))+0.3*x(1)*x(2);...
               2*x(1)*x(2)-5*x(2)*(1-x(1)-x(2))-2*x(2)*x(2)];
   [t1,x1]=ode45(fDae,[0,20],x0); plot(t1,x1,t1,1-sum(x1'))
```

注意，这里如果使用 ode45() 函数也不会出现求解错误。

利用隐式微分方程求解函数 ode15i()，则用匿名函数可以描述微分方程

```
>> f=@(t,x,xd)[xd(1)+0.2*x(1)-x(2)*x(3)-0.3*x(1)*x(2);...
       xd(2)-2*x(1)*x(2)+5*x(2)*x(3)+2*x(2)^2; x(1)+x(2)+x(3)-1];
```

令 $\boldsymbol{x}_0=[0.8,0.1,*]^{\mathrm{T}}$，其中 * 表示自由值，则可以用下面的语句解出相容的初始条件，并直接求解该微分代数方程，得出的结果将和前面完全一致，但求解更直观。

```
>> x0=[0.8;0.1;2]; x0F=[1;1;0]; xd0=[1;1;1]; xd0F=[];
   [x0,xd0]=decic(f,0,x0,x0F,xd0,xd0F),
   res=ode15i(f,[0,20],x0,xd0); plot(res.x,res.y)
```

得出状态变量初值 $\boldsymbol{x}_0=[0.8,0.1,0.1]^{\mathrm{T}}$，状态变量导数初值 $\dot{\boldsymbol{x}}_0=[-0.126,0.09,1]^{\mathrm{T}}$。

例 4-23 试用微分代数方程求解的方式求解例 4-19 中定义的隐式微分方程。

求解 在例 4-19 中采用对 $\boldsymbol{A}(\boldsymbol{x})$ 矩阵直接求逆的形式原隐式方程转换成显式一阶微分方程组，这样就可以用一般微分方程组数值解法直接得出方程的解。其实，在这样的求解过程作了一个假设为 $\boldsymbol{A}(\boldsymbol{x})$ 矩阵为非奇异矩阵，虽然对这个例子碰巧是正确的，但这种解法毕竟不严密，所以应该需要采用微分代数方程的方法来求解该问题。

对原方程进行分析发现，可以编写匿名函数来描述微分方程和质量矩阵 $\boldsymbol{A}(\boldsymbol{x})$

```
>> f=@(t,x)[1-x(1); -x(2)];
   fM=@(t,x)[sin(x(1)),cos(x(2)); -cos(x(2)),sin(x(1))];
```

注意，可能是 MATLAB 本身的漏洞，该函数不支持 inline() 形式定义的函数，只能采用 M-函数和匿名函数的形式。这样就可以用下面的语句调用微分代数方程求解微分代数方程。

```
>> options=odeset; options.Mass=fM; options.RelTol=1e-6;
   [t,x]=ode45(f,[0,10],[0;0],options); plot(t,x)
```

得出的图形仍将和图 4-14 中的曲线完全一致。

广义线性系统的状态方程模型可以写成

$$E\dot{x}(t) = Ax(t) + Bu(t) \tag{4-3-3}$$

其中 E 为奇异矩阵。有了前面介绍的微分代数方程的求解方法，对这样的广义系统仿真就轻而易举了。

例 4-24 考虑广义线性系统模型，设输入为 $u(t)=\mathrm{e}^{-0.5t}\sin(4t+2)$，且

$$E=\begin{bmatrix}0 & -1 & 0 & -1\\ 1 & 2 & 0 & 1\\ -2 & -2 & 1 & -2\\ -1 & -1 & 0 & 0\end{bmatrix},\ A=\begin{bmatrix}-2 & -2 & 0 & -1\\ -2 & -4 & -1 & -2\\ 0 & -1 & -7 & -1\\ -1 & -2 & -1 & -5\end{bmatrix},\ B=\begin{bmatrix}0\\ -5\\ 2\\ 4\end{bmatrix}$$

若系统的初始状态为零，试求出各个状态的时间曲线。

求解 可以用匿名函数描述已知的状态方程，注意，这里的常数矩阵 S 和 B 可以事先在工作空间中定义，这时就可以由下面的语句求解此微分代数方程，得出如图 4-17 所示的状态曲线。

```
>> A=[-2,-2,0,-1; -2,-4,-1,-2; 0,-1,-7,-1; -1,-2,-1,-5];
   B=[0; -5; 2; 4]; M=[0,-1,0,-1; 1,2,0,1; -2,-2,1,-2; -1,-1,0,0];
   f=@(t,x)A*x+B*exp(-0.5*t)*sin(4*t+2);
   fopt=odeset; fopt.Mass=M; x0=[0; 0; 0; 0];
   [t,x]=ode15s(f,[0,10],x0,fopt); plot(t,x)
```

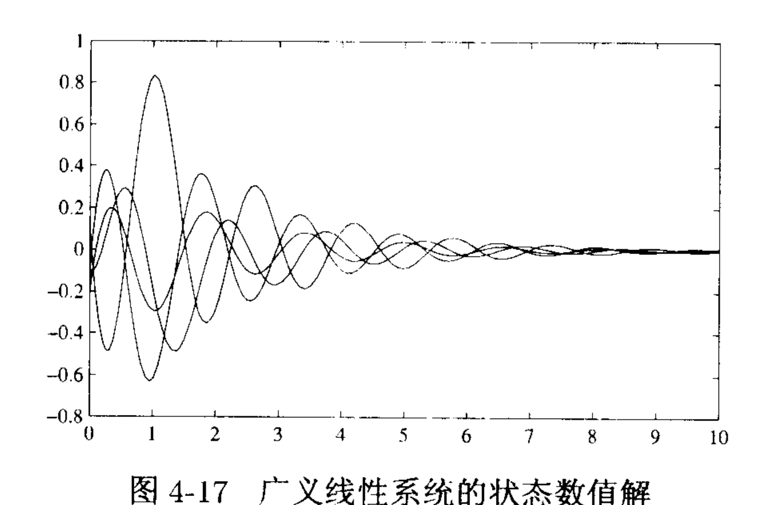


图 4-17 广义线性系统的状态数值解

4.3.4 延迟微分方程求解

在很多控制的分支中都需要研究一种带有时间延迟的数学模型，例如过程控制受控对象在受激励后一小段时间后才能开始响应，另外在网络控制中网络本身

存在的时间延迟等。延迟微分方程组的一般形式为

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(t,\boldsymbol{x}(t-\tau_1),\boldsymbol{x}(t-\tau_2),\cdots,\boldsymbol{x}(t-\tau_n)) \tag{4-3-4}$$

其中，$\tau_i \geqslant 0$ 为状态变量 $\boldsymbol{x}(t)$ 的延迟常数。

MATLAB 提供了求解这类方程的显式 Runge-Kutta 算法 `dde23()`，可以直接求解延迟微分方程。该函数的调用格式为

`sol=dde23(`f_1, $\boldsymbol{\tau}$, f_2, $[t_0,t_\mathrm{f}]$`)`

其中，$\boldsymbol{\tau}=[\tau_1,\tau_2,\cdots,\tau_n]$，$f_1$ 为描述延迟微分方程的 MATLAB 语言函数，f_2 为描述 $t\leqslant t_0$ 时的状态变量值的函数。如果是函数则可以为 MATLAB 语言函数，如果为常量则可以由向量直接给出。该函数返回的变量 `sol` 为结构体数据，其 `sol.x` 成员变量为时间向量 $\boldsymbol{t}$，成员变量 `sol.y` 为各个时刻的状态向量构成的矩阵 $\boldsymbol{x}$，和 `ode45()` 等返回的 $\boldsymbol{x}$ 矩阵是不一样的，它是按照行排列的，正好是该函数结果的转置矩阵。可见，该函数调用格式很不规范，期望能在后面的版本中有所改变。

例 4-25 假设已知延迟微分方程组为

$$\begin{cases}\dot{x}(t)=1-3x(t)-y(t-1)-0.2x^3(t-0.5)-x(t-0.5)\\ \ddot{y}(t)+3\dot{y}(t)+2y(t)=4x(t)\end{cases}$$

其中，在 $t\leqslant 0$ 时，$x(t)=y(t)=\dot{y}(t)=0$，试求出该方程的数值解。

求解 可见，该方程中含有 $x(t)$, $y(t)$ 信号在 $t,t-1,t-0.5$ 时刻的值，所以需要专门的延迟微分方程求解算法和程序来求解。若想得出该方程的数值解，需要将其变换成一阶显式微分方程组。实现转换的最直观方法是引入一组状态变量 $x_1(t)=x(t),x_2(t)=y(t),x_3(t)=\dot{y}(t)$，这样可以得出下面给出的一阶微分方程组

$$\begin{cases}\dot{x}_1(t)=1-3x_1(t)-x_2(t-1)-0.2x_1^3(t-0.5)-x_1(t-0.5)\\ \dot{x}_2(t)=x_3(t)\\ \dot{x}_3(t)=4x_1(t)-2x_2(t)-3x_3(t)\end{cases}$$

本方程可以定义两个时间常数 $\tau_1=1, \tau_2=0.5$。这样，由第一个方程可见，在其中需要 τ_1 延迟时间常数的是状态变量 x_2，而需要 τ_2 的状态变量是 x_1。由下面的匿名函数可以描述延迟微分方程，这样用下面的 MATLAB 语句可以立即得出该延迟微分方程的数值解。

```
>> f=@(t,x,z)[1-3*x(1)-z(2,1)-0.2*z(1,2)^3-z(1,2);...
              x(3); 4*x(1)-2*x(2)-3*x(3)];
   lags=[1 0.5]; tx=dde23(f,lags,zeros(3,1),[0,10]);
   plot(tx.x,tx.y(2,:))
```

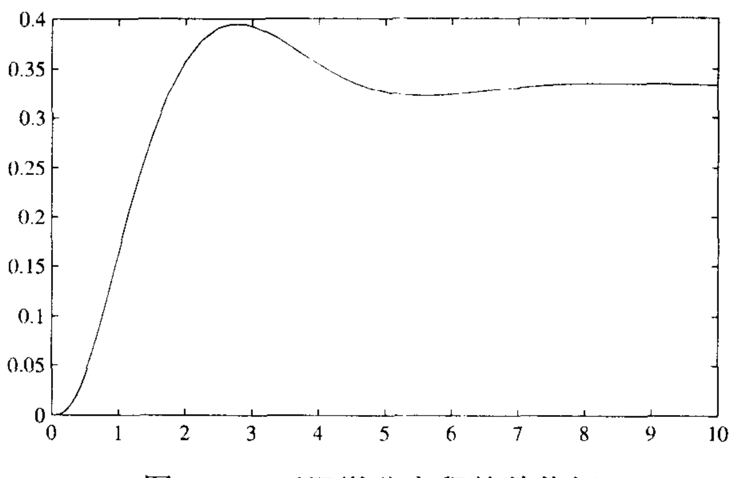


图 4-18 延迟微分方程的数值解

用下面的语句可以绘制出 $y(t)$ 信号，如图 4-18 所示。

注意，在描述延迟微分方程时，引入了矩阵 z。因为本例有两个延迟时间常数 τ_1 和 τ_2，所以 z 的第一列表示 $\boldsymbol{x}(t-\tau_1)$，第 2 列表示 $\boldsymbol{x}(t-\tau_2)$，这样 $x_1(t-0.5)$ 显然是发生在 τ_2 时刻，且为第 1 个状态变量的值，所以可以用 $z(1,2)$ 表示。

例 4-26 前面介绍了 dde23() 函数在求解延迟微分方程中的应用。然而对于如下问题

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_1\boldsymbol{x}(t-\tau_1) + \boldsymbol{A}_2\dot{\boldsymbol{x}}(t-\tau_2) + \boldsymbol{B}u(t)$$

其中，$\tau_1 = 0.15, \tau_2 = 0.5$，因为方程中同时包含 $\dot{x}(t)$ 和 $\dot{x}(t-\tau_2)$ 项，这类微分方程又称为中性 (neutral-type) 延迟微分方程，所以单纯采用 dde23() 是无能为力的，而需要引入更好的工具，如系统仿真的框图化求解环境 Simulink 来实现。详细的求解方法将在后面介绍。

4.3.5 多模型切换系统的求解

切换系统是控制理论中的一个重要的研究领域[5]，所谓切换系统就是在某种规律下其模型在多个模型间切换的系统。切换系统的数学模型可以表示为

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_i(t,\boldsymbol{x},\boldsymbol{u}),\ \ i=1,\cdots,m \tag{4-3-5}$$

该系统允许在某个控制规律下，整个系统在各个模型之间切换。利用切换系统的理论，可以设计控制器，使得不稳定的各个模型 $\boldsymbol{f}_i$ 通过合理的切换达到整个系统的稳定。

例 4-27 假设已知系统模型 $\dot{\boldsymbol{x}} = \boldsymbol{A}_i\boldsymbol{x}$，其中 $\boldsymbol{A}_1 = \begin{bmatrix} 0.1 & -1 \\ 2 & 0.1 \end{bmatrix}$，$\boldsymbol{A}_2 = \begin{bmatrix} 0.1 & -2 \\ 1 & 0.1 \end{bmatrix}$。可见，两个系统都不稳定。若 $x_1x_2 < 0$，即状态处于第 II, IV 象限时，切换到系统 $\boldsymbol{A}_1$，

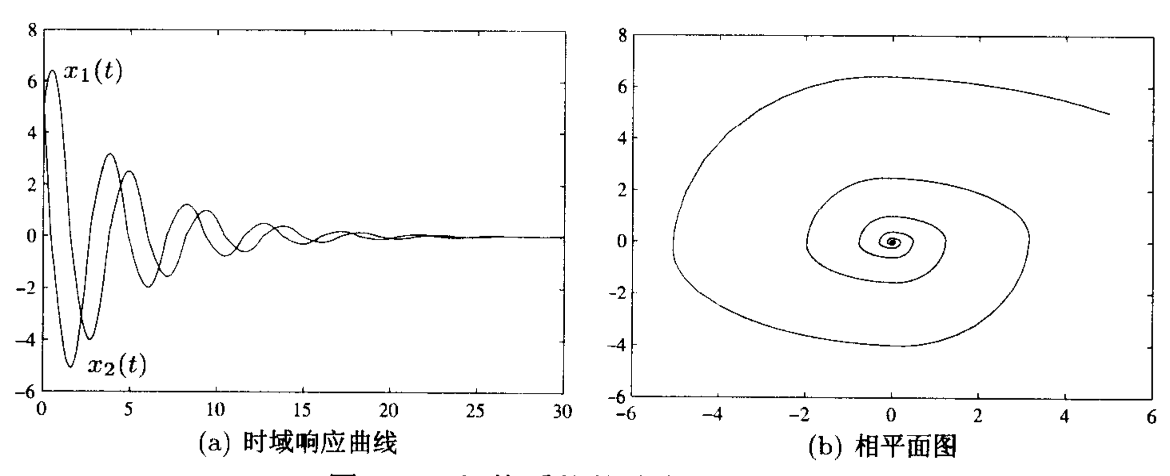
而 $x_1x_2 \geqslant 0$，即状态处于 I,III 象限时切换到 $\boldsymbol{A}_2$。令初始状态为 $x_1(0) = x_2(0) = 5$，试求解该系统的微分方程。

求解 按照系统模型及切换律，可以容易地写出切换系统的 MATLAB 表示为

```
function dx=switch_sys(t,x)
if x(1)*x(2)<0, A=[0.1 -1; 2 0.1];
else, A=[0.1 -2; 1 0.1]; end
dx=A*x;
```

这样就能用下面的语句直接求解切换系统的方程，得出如图 4-19 所示的时间响应曲线和相平面曲线。可见，不稳定的状态方程模型在某种指定的切换律下，可以得出稳定的整体系统状态。

```
>> [t,x]=ode45(@switch_sys,[0,30],[5;5]);
   plot(t,x), figure, plot(x(:,1),x(:,2))
```



(a) 时域响应曲线　　(b) 相平面图

图 4-19　切换系统的响应与切换效果

其实，为方便起见，该切换系统还可以由匿名函数表示成

```
>> f=@(t,x)(x(1)*x(2)<0)*[0.1 -1; 2 0.1]*x+...
          (x(1)*x(2)>=0)*[0.1 -2; 1 0.1]*x;
```

4.3.6　随机信号激励下线性微分方程的离散化求解

假设线性连续系统的状态方程模型为

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}[\boldsymbol{d}(t) + \gamma(t)],\ \ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{4-3-6}$$

式中 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 为兼容矩阵，$\boldsymbol{d}(t)$ 为确定性输入向量，$\gamma(t)$ 为 Gauss 白噪声向量，满足下式：

$$\mathrm{E}[\boldsymbol{\gamma}(t)] = 0,\ \ \mathrm{E}[\boldsymbol{\gamma}(t)\boldsymbol{\gamma}^{\mathrm{T}}(t)] = \boldsymbol{V}_\sigma\delta(t - \tau) \tag{4-3-7}$$

定义一个变量 $\gamma_c(t) = \boldsymbol{B}\gamma(t)$，则可以证明 $\gamma_c(t)$ 亦为 Gauss 白噪声，满足

$$\mathrm{E}[\gamma_c(t)] = 0,\ \ \mathrm{E}[\gamma_c(t)\gamma_c^{\mathrm{T}}(t)] = \boldsymbol{V}_c\delta(t-\tau) \tag{4-3-8}$$

其中 $\boldsymbol{V}_c = \boldsymbol{B}\boldsymbol{V}_\sigma\boldsymbol{B}^{\mathrm{T}}$ 为一个协方差矩阵，这时式 (4-3-6) 可以改写成

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{d}(t) + \gamma_c(t),\ \ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{4-3-9}$$

状态变量的解析解可以写成：

$$\boldsymbol{x}(t) = \mathrm{e}^{-\boldsymbol{A}t}\boldsymbol{x}(t_0) + \int_{t_0}^{t} \mathrm{e}^{\boldsymbol{A}(t-\tau)}d(\tau)\boldsymbol{B}\mathrm{d}\tau + \int_{t_0}^{t}\gamma_c(t)\mathrm{d}\tau \tag{4-3-10}$$

假设 $t_0 = k\Delta t$, $t = (k+1)\Delta t$，其中 Δt 为计算步长，并假定在一个计算步长之内确定性输入信号 $d(t)$ 为一个常数，亦即，如 $\Delta t \leqslant t \leqslant (k+1)\Delta t$ 时有 $\boldsymbol{d}(t) = \boldsymbol{d}(k\Delta t)$，则式 (4-3-10) 的离散形式可以写成

$$\boldsymbol{x}[(k+1)\Delta t] = \boldsymbol{F}\boldsymbol{x}(k\Delta t)+\boldsymbol{G}\boldsymbol{d}(k\Delta t)+\gamma_d(k\Delta t),\ \ \boldsymbol{y}(k\Delta t) = \boldsymbol{C}\boldsymbol{x}(k\Delta t) \tag{4-3-11}$$

式中 $\boldsymbol{F} = \mathrm{e}^{\boldsymbol{A}\Delta t}$, $\boldsymbol{G} = \displaystyle\int_0^{\Delta t} \mathrm{e}^{\boldsymbol{A}(\Delta t-\tau)}\boldsymbol{B}\mathrm{d}\tau$，且

$$\gamma_d(k\Delta t) = \int_{k\Delta t}^{(k+1)\Delta t} \mathrm{e}^{\boldsymbol{A}[(k+1)\Delta t-\tau]}\gamma_c(t)\mathrm{d}\tau = \int_0^{\Delta t}\mathrm{e}^{\boldsymbol{A}t}\gamma_c[(k+1)\Delta t-\tau]\mathrm{d}\tau \tag{4-3-12}$$

可见矩阵 $\boldsymbol{F}$, $\boldsymbol{G}$ 和确定性系统的离散化形式是一样的，所以会很容易求得，但可以看出，若系统含有随机输入时，系统的离散化形式与传统形式是不同的。可以证明 $\gamma_d(t)$ 亦为 Gauss 白噪声向量，且满足

$$\mathrm{E}[\gamma_d(k\Delta t)] = 0,\ \ \mathrm{E}[\gamma_d(k\Delta t)\gamma_d^{\mathrm{T}}(j\Delta t)] = \boldsymbol{V}\delta_{kj} \tag{4-3-13}$$

式中 $\boldsymbol{V} = \displaystyle\int_0^{\Delta t} \mathrm{e}^{\boldsymbol{A}t}\boldsymbol{V}_c\mathrm{e}^{\boldsymbol{A}^{\mathrm{T}}t}\mathrm{d}t$。利用 Taylor 级数展开技术可得

$$\boldsymbol{V} = \int_0^{\Delta t}\sum_{k=0}^{\infty}\frac{\boldsymbol{R}_k(0)}{k!}t^k\mathrm{d}t = \sum_{k=0}^{\infty}\boldsymbol{V}_k \tag{4-3-14}$$

其中 $\boldsymbol{R}_k(0)$ 与 $\boldsymbol{V}_k$ 可以由下式递推地求出[6]

$$\begin{cases} \boldsymbol{R}_k(0) = \boldsymbol{A}\boldsymbol{R}_{k-1}(0) + \boldsymbol{R}_{k-1}(0)\boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{V}_k = \dfrac{\Delta t}{k+1}(\boldsymbol{A}\boldsymbol{V}_{k-1} + \boldsymbol{V}_{k-1}\boldsymbol{A}^{\mathrm{T}}) \end{cases} \tag{4-3-15}$$

递推初值为 $\boldsymbol{R}_0(0)=\boldsymbol{R}(0)=\boldsymbol{V}_{\rm c}$, $\boldsymbol{V}_0=\boldsymbol{V}_{\rm c}\Delta t$。由奇异值分解理论，可以将矩阵 $\boldsymbol{V}$ 写成 $\boldsymbol{V}=\boldsymbol{U\Gamma U}^{\rm T}$，其中 $\boldsymbol{U}$ 为正交矩阵，$\boldsymbol{\Gamma}$ 为含有非零对角元素的对角矩阵，这样可以得出 Cholesky 分解 $\boldsymbol{V}=\boldsymbol{DD}^{\rm T}$。且 $\gamma_{\rm d}(k\Delta t)=\boldsymbol{De}(k\Delta t)$，式中 $\boldsymbol{e}(k\Delta t)$ 为 $n\times 1$ 向量，且 $\boldsymbol{e}(k\Delta t)=[e_k,e_{k+1},\cdots,e_{k+n-1}]^{\rm T}$，使得各个分量 e_k 满足标准正态分布，即 $e_k\sim N(0,1)$。系统的离散形式的递推解可以写成

$$\begin{cases}\boldsymbol{x}[(k+1)\Delta t]=\boldsymbol{Fx}(k\Delta t)+\boldsymbol{Gd}(k\Delta t)+\boldsymbol{De}(k\Delta t)\\ \boldsymbol{y}(k\Delta t)=\boldsymbol{Cx}(k\Delta t)\end{cases}\tag{4-3-16}$$

根据上面的算法，可以编写出随机输入下连续线性系统离散化的 MATLAB 函数 sc2d() 如下：

```
function [F,G,D,C]=sc2d(G,sig,T)
G=ss(G); G=balreal(G); Gd=c2d(G,T); A=G.a; B=G.b; C=G.c; i=1;
F=Gd.a; G=Gd.b; V0=B*sig*B'*T; Vd=V0; V1=Vd;
while (norm(V1)<eps)
   V1=T/(i+1)*(A*V0+V0*A'); Vd=Vd+V1; V0=V1; i=i+1;
end
[U,S,V0]=svd(Vd); V0=sqrt(diag(S)); Vd=diag(V0); D=U*Vd;
```

该函数的调用格式为 [$\boldsymbol{F}$,$\boldsymbol{G}$,$\boldsymbol{D}$,$\boldsymbol{C}$]=sc2d(G,σ,Δt)，其中，G 为系统模型，σ 为输入信号协方差矩阵，Δt 为采样周期，$(\boldsymbol{F},\boldsymbol{G},\boldsymbol{D},\boldsymbol{C})$ 为离散化状态方程的相应矩阵。

在仿真时，可以产生一组伪随机数，从而产生向量 $\boldsymbol{e}(k\Delta t)$，然后求出状态变量 $\boldsymbol{x}[(k+1)\Delta t]$ 并求出输出变量 $y[(k+1)\Delta t]$。

例 4-28 考虑受控对象的传递函数模型为 $G(s)=\dfrac{s^3+7s^2+24s+24}{s^4+10s^3+35s^2+50s+24}$，如果用白噪声信号激励该系统，试对其进行仿真分析并得出输出信号的统计规律。

求解 假设系统的采样周期为 $T=0.02$，下面的语句

```
>> G=tf([1,7,24,24],[1,10,35,50,24]); [F,G0,D,C]=sc2d(G,1,0.02)
```

可以得出离散化的状态方程模型为

$$\boldsymbol{F}=\begin{bmatrix}0.9838 & -0.00673 & 0.0132 & 0.00129\\ 0.00673 & 0.9883 & 0.07022 & 0.00364\\ 0.0132 & -0.07022 & 0.8653 & -0.0257\\ 0.00129 & -0.0036401 & -0.0257 & 0.9684\end{bmatrix},\quad \boldsymbol{G}_0=\begin{bmatrix}0.01823\\ -0.00355\\ -0.00757\\ -0.000718\end{bmatrix}$$

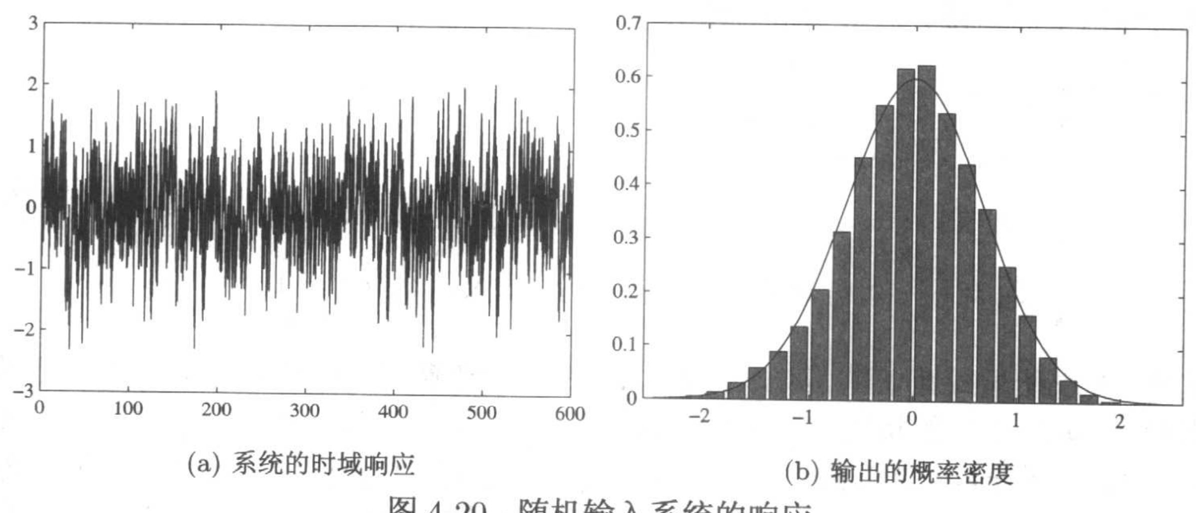
$$\boldsymbol{D}=\begin{bmatrix}-0.12893 & 0.00088028 & -4.6919\times10^{-6} & 4.6917\times10^{-10}\\ 0.0251 & -0.0012 & -1.3573\times10^{-5} & 2.3791\times10^{-9}\\ 0.05356 & 0.002635 & -5.2322\times10^{-6} & -8.8812\times10^{-10}\\ 0.00508 & 0.0005 & 3.1358\times10^{-6} & 9.5176\times10^{-9}\end{bmatrix}$$

由离散化的状态方程模型出发，可以用下列的 MATLAB 语句对之进行仿真，其中仿真点数设为 30000个。

```
>> n=30000; e=randn(n+4,1); e=e-mean(e);  % 生成零均值的输入信号
   y=zeros(n,1); x=zeros(4,1); d0=0;
   for i=1:n, x=F*x+G0*d0+D*e(i:i+3); y(i)=C*x; end
   T=0.02; t=0:T:(n-1)*T; plot(t,y), v=norm(G)
```

得出的响应曲线如图 4-20(a) 所示，同时还可以得出系统的 $\mathcal{H}_2$ 范数的值为 $v = 0.6655$。不过从得出的曲线看，这样的响应似乎杂乱无章，所以对随机输入来说，分析其统计规律应该更有用。可以考虑将输出范围 $(-2.5, 2.5)$ 划分成宽度为 $w = 0.2$ 的小区间，累加出落入每个小区间的输出点个数，由这些值除以 nw 则可以得出基于仿真结果的概率密度值，如图 4-20 (b) 所示。另外，可以从理论上证明[7]，输出信号的概率密度为 $p(y) = \dfrac{1}{\sqrt{2\pi}v}\mathrm{e}^{-y^2/(2v^2)}$，这样，在得出的直方图上还可以叠印上系统的理论概率密度，可见和由仿真得出的结果较吻合。

```
>> w=0.2; x=-2.5:w:2.5; y1=hist(y,x); bar(x,y1/n/w);
   x1=-2.5:0.05:2.5; y2=1/sqrt(2*pi)/v*exp(-x1.^2/2/v^2);
   line(x1,y2)  % 叠印理论概率密度函数曲线
```



(a) 系统的时域响应　　(b) 输出的概率密度

图 4-20　随机输入系统的响应

4.4 微分方程边值问题的计算机求解

前面的微分方程数值解中侧重研究初值问题，即已知 $\boldsymbol{x}_0$ 对其他时刻状态变量值进行求解的方法。在实际应用中，经常会遇到这样的问题，已知部分状态在 $t = 0$ 时刻的值，还知道部分状态在 $t = t_\mathrm{f}$ 时刻的值，这类问题即所谓的边值问题。边值问题也是 `ode45()` 类函数无法直接求解的一类问题。本节将讨论边值问题的计算机求解方法，并以此为基础介绍边值问题在线性二次型最优控制仿真中的应用。

4.4.1　二阶微分方程两点边值问题的求解方法

二阶微分方程的边值问题的数学描述为

$$\ddot{y}(t)=F(t,y,\dot{y}),\ \ y(a)=\gamma_{\rm a},\ y(b)=\gamma_{\rm b} \tag{4-4-1}$$

假设想在区间 $[a,b]$ 上研究该方程的解，且已知在这两个边界点上的值，则上面的二阶微分方程称为两点边值问题。

显然，利用前面的初值问题算法是不能直接使用的，因为并不能直接获得在初始时刻的各个变量的值。假定原始问题可以转换成下面的初值问题

$$\ddot{y}(t)=F(t,y,\dot{y}),\ \ y(a)=\gamma_{\rm a},\ \dot{y}(a)=m \tag{4-4-2}$$

则问题转换成求解 $y(m;b)=\gamma_{\rm b}$，可以采用下面的 Newton 迭代法来求取 m。

$$m_{i+1}=m_i-\frac{y(m_i;b)-\gamma_{\rm b}}{(\partial y/\partial m)(m_i;b)}=m_i-\frac{v_1(b)-\gamma_{\rm b}}{v_3(b)} \tag{4-4-3}$$

式中，$v_1=y(m_i;t), v_2=\dot{y}(m_i;t), v_3=(\partial y/\partial m)(m_i;t), v_4=(\partial\dot{y}/\partial m)(m_i;t)$，且可以由下面的微分方程初值问题来求解。

$$\begin{cases}\dot{v}_1=v_2, & v_1(a)=\gamma_{\rm a}\\ \dot{v}_2=F(t,v_1,v_2), & v_2(a)=m\\ \dot{v}_3=v_4, & v_3(a)=0\\ \dot{v}_4=\dfrac{\partial F}{\partial y}(t,v_1,v_2)v_3+\dfrac{\partial F}{\partial \dot{y}}(t,v_1,v_2)v_4, & v_4(a)=1\end{cases} \tag{4-4-4}$$

其中，要求能显式地求出 $\partial F/\partial y, \partial F/\partial\dot{y}$。在具体计算中可以指定一个 m 值，然后求解式 (4-4-4) 中的初值问题，将结果代入式 (4-4-3) 迭代一步，并将结果代入式 (4-4-4) 重新计算，直至两次计算出来的 m 值的误差在允许的范围内，则可采用此 m 值，这样的迭代方法又称为非线性边值问题的打靶算法。最后代入式 (4-4-2) 来求解原始问题。上面算法的 MATLAB 实现为

```
function [t,y]=nlbound(funcn,funcv,tspan,x0f,tol,varargin)
t0=tspan(1);tfinal=tspan(2);  ga=x0f(1); gb=x0f(2); m=1; m0=0;
while (norm(m-m0)>tol), m0=m;
   [t,v]=ode45(funcv,tspan,[ga;m;0;1],varargin);
   m=m0-(v(end,1)-gb)/(v(end,3));
end
[t,y]=ode45(funcn,tspan,[ga;m],varargin);
```

其中，用户必须自己编写一个 funcv() 函数来描述式 (4-4-4) 中的初值问题。下面将通过例子来演示此算法。

例 4-29 试求解下面的非线性微分方程边值问题。

$$\ddot{y} = F(x, y, \dot{y}) = 2y\dot{y},\ y(0) = -1,\ y(\pi/2) = 1$$

求解 可以容易地求出偏导数 $\partial F/\partial y = 2\dot{y}$, $\partial F/\partial \dot{y} = 2y$。令 $x_1 = y, x_2 = \dot{y}$, 代入式 (4-4-4) 中的第 4 个式子可以立即得出 $\dot{x}_4 = 2x_2x_3 + 2x_1x_4$, 故可以写出下面的 MATLAB 语句来求解微分方程的边值问题。

```
>> f=@(t,x)[x(2); 2*x(1)*x(2); x(4); 2*x(2)*x(3)+2*x(1)*x(4);
   g=@(t,x)[x(2); 2*x(1)*x(2)];
   [t,y]=nlbound(g,f,[0,pi/2],[-1,1],1e-8);
```

4.4.2 一般边值微分方程的求解方法

由前面的叙述可见，两点边值求解方法的局限性较大，对原来的方程有许多约束，如阶次约束、边值约束等。如果需要求解更复杂的边值问题，必须利用其他的方法。假设要研究的微分方程为

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{\theta}) \tag{4-4-5}$$

其中 $\boldsymbol{y}$ 为状态变量向量，$\boldsymbol{\theta}$ 为方程中其他未知参数向量。该方程已知的边界值为

$$\boldsymbol{\phi}[\boldsymbol{y}(a), \boldsymbol{y}(b), \boldsymbol{\theta}] = \mathbf{0} \tag{4-4-6}$$

如果想将原始问题变换成初值问题，则需要求解若干个代数方程。若需要求解的变量个数和这样建立起来的方程个数相同时，则可以通过求解方程的方法先将它们求解出来，然后求解微分方程。和典型的边值问题相比，这里研究的方程求解更具一般性，因为除了传统的边值问题之外，还可以求解其他的未知参数问题。

MATLAB 提供的 bvp4c() 函数[8]可以很好地求解微分方程的边值问题。正确求解一个常微分方程的边值问题，一般应该经过以下几个步骤:

① **参数初始化** 调用 bvpinit() 函数即可输入信息。当然这样的描述决不仅仅局限于边值，其他待定变量也可以在这里一起描述。该函数的调用格式为 sinit=bvpinit($\boldsymbol{v}$,$\boldsymbol{x}_0$,$\boldsymbol{\theta}_0$)，其中，v 应该包含测试的时间向量，可以用 $\boldsymbol{v}$=linspace(a,b,M) 生成，注意为保障足够快的计算速度，M 不能取得过大，一般取 $M = 5$ 即可。除了 $\boldsymbol{v}$ 向量，当然还应该给出状态变量初值 $\boldsymbol{x}_0$ 和待定参数 $\boldsymbol{\theta}_0$ 的初始搜索点。

② **微分方程和边值问题的 MATLAB 函数描述** 微分方程本身的描述和初值问题完全一致，边值问题描述出式 (4-4-6) 中的各个式子即可，具体格式将由下面的例子演示。

③ **边值问题的求解** 调用 bvp4c() 函数就可以直接求解边值问题了

sol=bvp4c('fun1','fun2',sinit,options,附加参数)

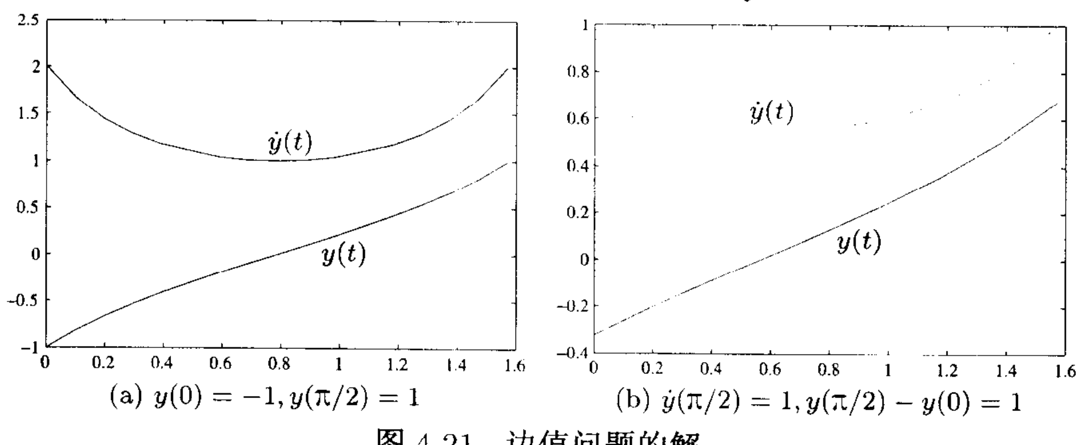
其中，fun1.m 和 fun2.m 分别为描述微分方程和边值条件的 MATLAB 函数，当然它们也可以通过 inline() 函数或匿名函数直接表示。返回的 sol 为结构体型变量，其 sol.x 为 t 向量，sol.y 的每一行对应一个状态变量。sol.parameters 将返回求解出来的待定参数 $\boldsymbol{\theta}$。

后面将通过例子介绍该函数的编写方法。另外，还需要编写一个函数来描述一阶微分方程组，这和以前微分方程组的描述是完全一致的。

例 4-30 试用 bvp4c() 函数重新求解例 4-29 中的边值问题。

求解 令 $x_1 = y, x_2 = \dot{y}$，则可以得出一阶显式微分方程为 $\dot{x}_1 = x_2, \dot{x}_2 = 2x_1x_2$，这里采用了匿名函数的形式描述微分方程和边值条件，其效果和编写 MATLAB 函数是完全一致的。可以由下面的语句直接求解本边值问题，结果曲线如图 4-21 (a) 所示。得出的结果和前面例子是一致的。

```
>> f1=@(t,x)[x(2); 2*x(1)*x(2)]; f2=@(xa,xb)[xa(1)+1; xb(1)-1];
   sinit=bvpinit(linspace(0,pi/2,5),rand(2,1));
   sol=bvp4c(f1,f2,sinit); plot(sol.x,sol.y)
```



(a) $y(0) = -1, y(\pi/2) = 1$　　(b) $\dot{y}(\pi/2) = 1, y(\pi/2) - y(0) = 1$

图 4-21 边值问题的解

利用 bvp4c() 函数还可以解决更复杂的边值问题，例如若边值问题修改成 $\dot{y}(\pi/2) = 1$，$y(\pi/2) - y(0) = 1$，则可以如下修改 f2，并求解该方程，得出如图 4-21 (b) 所示的结果。

```
>> f2=@(xa,xb)[xb(2)-1; xb(1)-xa(1)-1];
   sol=bvp4c(f1,f2,sinit); plot(sol.x,sol.y)
```

例 4-31 已知某常微分方程模型为 $\begin{cases} \dot{x} = 4x - \alpha xy \\ \dot{y} = -2y + \beta xy \end{cases}$，且已知 $x(0) = 2, y(0) = 1$，$x(3) = 4, y(3) = 2$，试求出 α, β 并求解本微分方程。

求解 先引入状态变量 $x_1 = x, x_2 = y$，则可以将原问题转换成关于 $\boldsymbol{x}$ 的微分方程。另

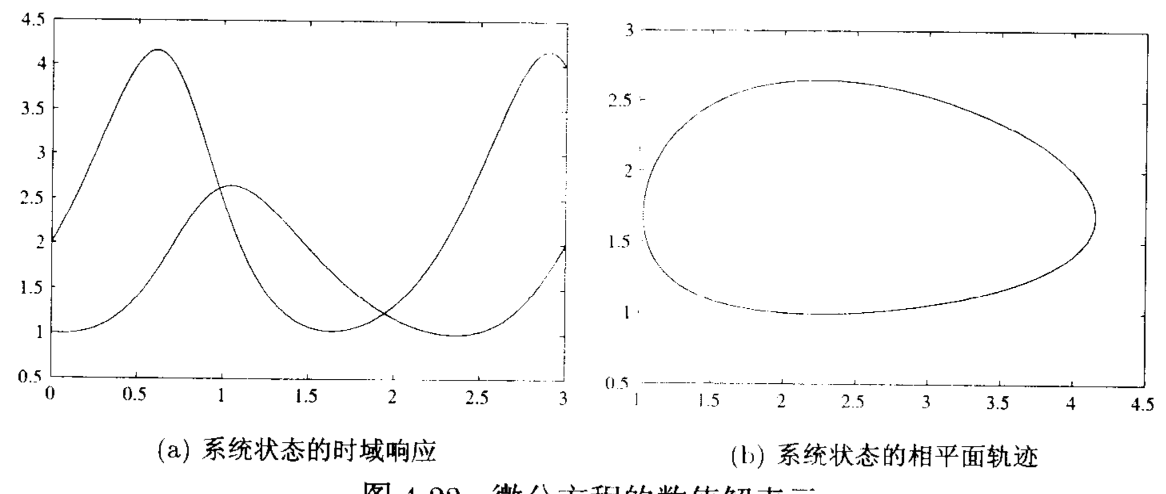
外，令 $v_1 = \alpha$，$v_2 = \beta$。和边值问题一样，需要描述系统的动态模型和边值问题如下

```
>> f=@(t,x,v)[4*x(1)+v(1)*x(1)*x(2); -2*x(2)+v(2)*x(1)*x(2)];
   g=@(ya,yb,v)[ya(1)-2; ya(2)-1; yb(1)-4; yb(2)-2];
```

从表示的函数可以看出，边值的描述还是很直观的。这时，可以先调用 bvpinit() 初始化函数来定义求解时间段及网格划分方法，并令状态初始值和参数 α,β 的初始值，因为有两个初始状态，两个未定参数，所以它们的初值均可以设置为随机数向量 rand(2,1)。定义了这些参数，则可以调用 bvp4c() 函数来求解边值问题的 α,β 参数，并求解在此参数下的系统方程了。

```
>> x1=[1;1]; x2=[-1;1]; sinit=bvpinit(linspace(0,3,5),x1,x2);
   sol=bvp4c(f,g,sinit); sol.parameters    % 显示待定参数
   plot(sol.x,sol.y); figure; plot(sol.y(1,:),sol.y(2,:));
```

得出的结果如图 4-22 所示，同时可以求出 $\alpha = -2.3721, \beta = 0.8934$。由得出的仿真曲线可以看出，方程状态的边值条件可以满足，所以求出的解是正确的。这里的初值向量 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 选择应该注意，如果选择不当可能使得求解过程中的 Jacobi 矩阵奇异，所以实际求解时若出现此现象，则应该选择其他的初值。



(a) 系统状态的时域响应　　(b) 系统状态的相平面轨迹

图 4-22 微分方程的数值解表示

4.4.3 微分 Riccati 方程与二次型最优控制问题求解

微分 Riccati 方程的一般数学表示方法

$$-\dot{\boldsymbol{P}}(t) = \boldsymbol{P}(t)\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}(t) + \boldsymbol{Q} - \boldsymbol{P}(t)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}(t) \tag{4-4-7}$$

其中，$\boldsymbol{P}(t_1) = \boldsymbol{P}_1$，$t \in [t_0, t_1]$。单从微分 Riccati 方程本身看，该问题并不是边值问题，而是终值问题，可以由 ode45() 这类函数直接求解。但考虑到在控制领域经常将微分 Riccati 方程和系统状态反馈相结合，而状态方程又涉及到初值模型，

所以这样需要综合考虑初值与终值问题，故可以将原始问题纳入边值问题的框架下求解。

这里先考虑简单微分 Riccati 方程的解析解方法，然后探讨一般微分 Riccati 方程的数值解法，最后将研究线性二次型最优控制问题的仿真方法。

1. 微分 Riccati 方程的解析解算法

假设 $\boldsymbol{P}(t)$ 为 $n\times n$ 方阵，这样可以构造出一个 $2n\times 2n$ 的微分方程

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{X}(t)\\ \boldsymbol{Y}(t)\end{bmatrix}=\begin{bmatrix}\boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\\ -\boldsymbol{Q} & -\boldsymbol{A}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}(t)\\ \boldsymbol{Y}(t)\end{bmatrix},\ \text{终值}\begin{bmatrix}\boldsymbol{X}(t_1)\\ \boldsymbol{Y}(t_1)\end{bmatrix}=\begin{bmatrix}\boldsymbol{I}_n\\ \boldsymbol{P}_1\end{bmatrix}\tag{4-4-8}$$

其中 $\boldsymbol{X}(t)$ 和 $\boldsymbol{Y}(t)$ 均为 $n\times n$ 矩阵，这样方程的结构和以前研究的一阶显式微分方程组 (4-2-1) 是不同的。若在 $t\in[t_0,t_1]$ 区间内方程的解矩阵 $\boldsymbol{X}(t)$ 非奇异，则微分 Riccati 方程的解析解可以写成[9]

$$\boldsymbol{P}(t)=\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(t)\tag{4-4-9}$$

可以证明，若 Hamilton 矩阵 $\boldsymbol{M}=\begin{bmatrix}\boldsymbol{A} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\\ -\boldsymbol{Q} & -\boldsymbol{A}^{\mathrm{T}}\end{bmatrix}$ 没有纯虚数特征值，且若 λ 是该矩阵的特征值，则 $-\lambda$ 也是该矩阵的特征值。可以引入相似变换矩阵 $\boldsymbol{T}=\begin{bmatrix}\boldsymbol{T}_{11} & \boldsymbol{T}_{12}\\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22}\end{bmatrix}$，则 $\begin{bmatrix}\widehat{\boldsymbol{X}}\\ \widehat{\boldsymbol{Y}}\end{bmatrix}=\boldsymbol{T}^{-1}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}$，从而得 $\boldsymbol{T}^{-1}\boldsymbol{M}\boldsymbol{T}=\begin{bmatrix}-\boldsymbol{\varLambda} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\varLambda}\end{bmatrix}$。这时

$$\begin{bmatrix}\dot{\widehat{\boldsymbol{X}}}\\ \dot{\widehat{\boldsymbol{Y}}}\end{bmatrix}=\boldsymbol{T}^{-1}\boldsymbol{M}\boldsymbol{T}\begin{bmatrix}\widehat{\boldsymbol{X}}\\ \widehat{\boldsymbol{Y}}\end{bmatrix}=\begin{bmatrix}-\boldsymbol{\varLambda} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\varLambda}\end{bmatrix}\begin{bmatrix}\widehat{\boldsymbol{X}}\\ \widehat{\boldsymbol{Y}}\end{bmatrix},\ \boldsymbol{T}\begin{bmatrix}\widehat{\boldsymbol{X}}(0)\\ \widehat{\boldsymbol{Y}}(0)\end{bmatrix}=\begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{P}_1\end{bmatrix}\tag{4-4-10}$$

这样可以得出

$$-\left[\boldsymbol{T}_{21}-\boldsymbol{P}_1\boldsymbol{T}_{11}\right]\widehat{\boldsymbol{X}}(0)=\left[\boldsymbol{T}_{22}-\boldsymbol{P}_1\boldsymbol{T}_{12}\right]\widehat{\boldsymbol{Y}}(0)\tag{4-4-11}$$

进而得出微分 Riccati 方程的解析解为

$$\boldsymbol{P}(t)=\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(t)=\left[\boldsymbol{T}_{21}+\boldsymbol{T}_{22}\mathrm{e}^{\boldsymbol{\varLambda}t}\boldsymbol{R}_{\mathrm{a}}\mathrm{e}^{\boldsymbol{\varLambda}t}\right]\left[\boldsymbol{T}_{11}+\boldsymbol{T}_{12}\mathrm{e}^{\boldsymbol{\varLambda}t}\boldsymbol{R}_{\mathrm{a}}\mathrm{e}^{\boldsymbol{\varLambda}t}\right]^{-1}\tag{4-4-12}$$

式中

$$\boldsymbol{R}_{\mathrm{a}}=-\left[\boldsymbol{T}_{22}-\boldsymbol{P}_1\boldsymbol{T}_{12}\right]^{-1}\left[\boldsymbol{T}_{21}-\boldsymbol{P}_1\boldsymbol{T}_{11}\right]\tag{4-4-13}$$

例 4-32 假设某系统矩阵、加权矩阵和初值矩阵分别为

$$\boldsymbol{A}=\begin{bmatrix}1 & 1\\ -6 & -4\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1\\ 2\end{bmatrix},\ \boldsymbol{Q}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix},\ R=1,\ \boldsymbol{P}_1(0)=\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}$$

试求解微分 Riccati 方程的解析解，并检验结果的正确性。

求解 由前面给出的算法，则用下面的语句可以立即得出原 Riccati 微分方程的解析解

```
>> A=[1,1; -6,-4]; B=[1; 2]; Q=zeros(2); P1=diag([1 2]); R=1;
   M=[A -B*inv(R)*B'; -Q -A']; [v,j]=jordan(sym(M));
   T1=eye(2); T1(3:4,3:4)=fliplr(T1); T=v*T1;
   T11=T(1:2,1:2); T12=T(1:2,3:4); T21=T(3:4,1:2); T22=T(3:4,3:4);
   Ra=-inv(T22-P1*T12)*(T21-P1*T11);
   syms t; Lam=j(3:4,3:4); EA=expm(Lam*t);
   P=simple((T21+T22*EA*Ra*EA)*inv(T11+T12*EA*Ra*EA))
```

得出的解析解为

$$\begin{cases} p_{11}(t) = -\dfrac{18\mathrm{e}^{2t}\left(176\mathrm{e}^{2t} - 12\mathrm{e}^{4t} - 316\mathrm{e}^{t} + 153\right)}{-491 - 6320\mathrm{e}^{3t} + 3825\mathrm{e}^{2t} + 3168\mathrm{e}^{4t} - 200\mathrm{e}^{6t}} \\ p_{12}(t) = p_{21}(t) = -\dfrac{6\mathrm{e}^{2t}\left(264\mathrm{e}^{2t} - 22\mathrm{e}^{4t} - 395\mathrm{e}^{t} + 153\right)}{-491 - 6320\mathrm{e}^{3t} + 3825\mathrm{e}^{2t} + 3168\mathrm{e}^{4t} - 200\mathrm{e}^{6t}} \\ p_{22}(t) = -\dfrac{6\mathrm{e}^{2t}\left(132\mathrm{e}^{2t} - 19\mathrm{e}^{4t} - 158\mathrm{e}^{t} + 51\right)}{-491 - 6320\mathrm{e}^{3t} + 3825\mathrm{e}^{2t} + 3168\mathrm{e}^{4t} - 200\mathrm{e}^{6t}} \end{cases}$$

下面的语句可以检验结果的正确性，看见得出的解是正确的。

```
>> subs(P,t,0), simple(diff(P)+P*A+A'*P+Q-P*B*inv(R)*B'*P)
```

2. 微分 Riccati 方程的数值解法

从前面的介绍可以看出，微分 Riccati 方程是可以求解析解的，然而，若系统的阶次增高，甚至问题稍变复杂，则上述的微分 Riccati 方程求解将变得异常困难，大部分问题的解析求解将变得不可能，所以在这样的情况下，应该考虑微分 Riccati 方程的数值解法。

可以编写一个通用的描述微分 Riccati 方程的 MATLAB 函数

```
function dy=diff_ric(t,x,flag,A,B,Q,R)
P=reshape(x,size(A)); Y=-P*A-A'*P-Q+P*B*inv(R)*B'*P; dy=Y(:);
```

这样用下面的语句就可以调用 ode45() 函数来求解微分 Riccati 方程

$[\boldsymbol{t},\boldsymbol{p}]$=ode45(@diff_ric,[$t_1$,0],$\boldsymbol{P}_1$(:),options,$\boldsymbol{A},\boldsymbol{B},\boldsymbol{Q},\boldsymbol{R}$)

注意，在该方程求解中，终止仿真时间可以小于初始仿真时间 t_1。

例 4-33 假设系统模型、加权和初值分别如下给出

$$\boldsymbol{A} = \begin{bmatrix} -6 & -6 & -17 \\ -1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix},\ \boldsymbol{Q} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix},\ R = 1,\ \boldsymbol{P}_1(0.5) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

试求解微分 Riccati 方程。

求解 由于系统的阶次增加，用前面介绍的解析方法是不可能得出问题的解的，所以必须借助数值方法来求解。描述了已知矩阵，则可以由下面语句求出方程的数值解，并绘制出 $\boldsymbol{P}$ 矩阵的曲线，如图 4-23 所示。

```
>> A=[-6,-6,-17; -1,0,1; 1,0,0]; B=[0; 2; 1];
   Q=[1 2 0; 2 8 0; 0 0 4]; R=1; P1=diag([1,3,8]);
   [t,y]=ode45(@diff_ric,[0.5,0],P1(:),[],A,B,Q,R); plot(t,y)
```

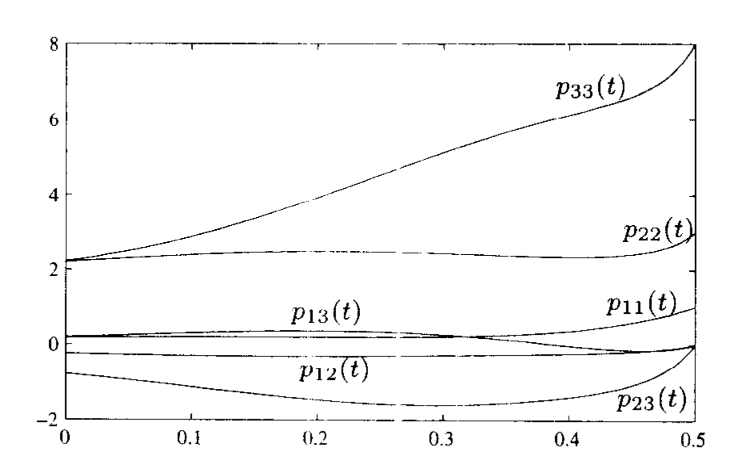


图 4-23 微分 Riccati 方程的数值解

由得出的数据可见，由已知的 $\boldsymbol{P}_1(0.5)$ 可以反推回 $\boldsymbol{P}_1(0)$ 矩阵。以该矩阵为初值，还可以用下面的语句求解微分方程的初值问题，得出和图 4-23 中一致的曲线。

```
>> P0=y(end,:);
   [t,y]=ode45(@diff_ric,[0 0.5],P0(:),[],A,B,Q,R); plot(t,y)
```

值得指出的是，虽然对上面的例子来说，可以解出初值，然后用初值可以反推回终值，但很多情况下这样的反推不一定成功，因为如果 t_1 较大，则各个状态在 $t=0$ 之前很久就达到稳态值，这样反推得出的结果就不惟一了。

3. 微分 Riccati 方程在控制理论中的应用

考虑式 (3-4-28) ～ 式 (3-4-31) 中介绍的二次型指标最优控制问题。由最优控制量 $\boldsymbol{u}^*(t)=-\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(t)$ 可以直接构造出下面的微分方程边值问题。

$$\begin{cases} \dot{\boldsymbol{x}}(t)=(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P})\boldsymbol{x}(t), & \boldsymbol{x}(0)=\boldsymbol{x}_0 \\ \dot{\boldsymbol{P}}(t)=-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}(t)-\boldsymbol{P}(t)\boldsymbol{A}+\boldsymbol{P}(t)\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}(t)-\boldsymbol{Q}, & \boldsymbol{P}(t_1)=\boldsymbol{P}_1 \end{cases} \tag{4-4-14}$$

可见，在得出的模型中，其中部分状态变量给出了初值，而另一些给出了终值，所以这样的问题是微分方程的边值问题。

这样，令新状态变量 $\boldsymbol{y}=\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{P}(:)\end{bmatrix}$，其中 $\boldsymbol{P}(:)$ 是矩阵 $\boldsymbol{P}$ 按列展开构成的列向量，这时可以写出如下的 M 函数来描述本问题的一阶微分方程组

```
function dydt=dynamic_ric(t,y,A,B,Q,R,x0,P1)
n=length(B); P=reshape(y(n+1:end),n,n);
dydt=[(A-B*inv(R)*B'*P)*y(1:n);
      reshape(-A'*P-P*A+P*B*inv(R)*B'*P-Q,n^2,1)];
```

另外，描述边值的 M 函数也可以很容易建立起来，如下所示

```
function res=bv_ric(ya,yb,A,B,Q,R,x0,P1)
n=length(x0); res=[ya(1:n)-x0; yb(n+1:end)-P1(:)];
```

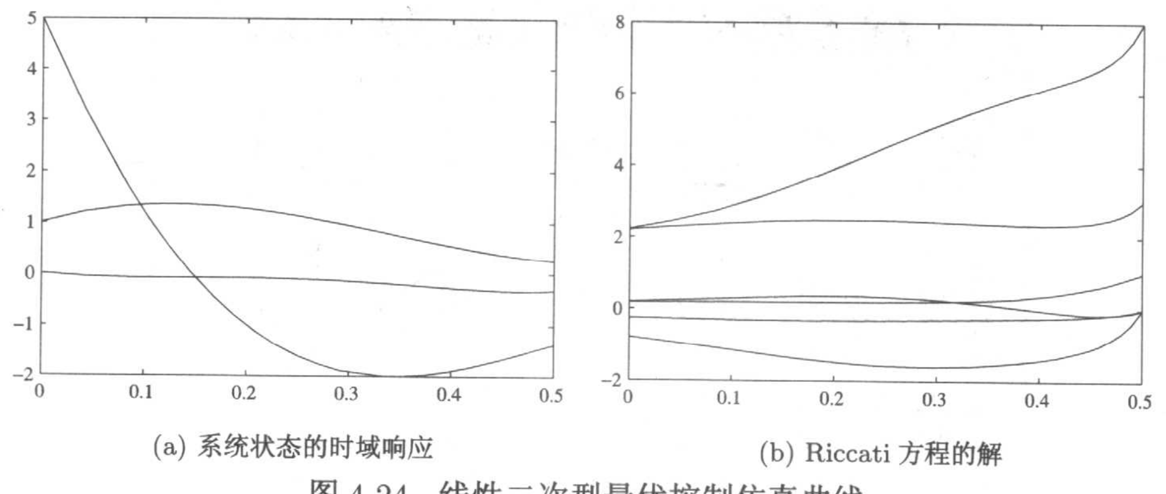
在这两个函数中，变量 $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{x}_0$ 和 $\boldsymbol{P}_1$ 为附加变量，可以根据实际系统的参数在 MATLAB 工作空间中直接定义。这时，可以由下面的语句求解二次型指标的最优控制问题。

sinit=bvpinit(linspace(a,b,M),rand(n^2+n,1))　% 构造初值
sol=bvp4c(@dynamic_ric,@bv_ric,sinit,options,...
$\boldsymbol{A},\boldsymbol{B},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{x}_0,\boldsymbol{P}_1$)　% 求解二次型最优控制问题

其中，M 是求解代数方程用的分段值，可以取为 5。返回的 sol 是结构体数据，sol.$\boldsymbol{x}$ 为时间向量，sol.$\boldsymbol{y}$ 为整个系统的状态向量在各个时刻构成的矩阵，其前 n 行数值是系统的状态 $\boldsymbol{x}$ 的仿真结果，后 n^2 行为 Riccati 方程的动态解。

例 4-34 仍考虑例 4-33 中给出的线性系统模型，假设状态变量的初值为 $\boldsymbol{x}_0 = [5,0,1]^{\mathrm{T}}$，试仿真 $[0,0.5]$ 区间的二次型最优控制系统。

求解 先输入系统已知的各个矩阵，然后可以用上面给出的方法定义初值变量 sinit，将其代入 bvp4c() 函数就可以直接求解式 (4-4-14) 中的边值问题了。这样得出的状态变量曲线如图 4-24 (a) 所示，微分 Riccati 方程的解在图 4-24 (b) 中给出，其结果与例 4-33 完全一致。



(a) 系统状态的时域响应　　(b) Riccati 方程的解

图 4-24　线性二次型最优控制仿真曲线

```
>> A=[-6,-6,-17; -1,0,1; 1,0,0]; B=[0; 2; 1];
```

```
Q=[1 2 0; 2 8 0; 0 0 4]; R=1; x0=[5;0;1]; P1=diag([1,3,8]);
sinit=bvpinit(linspace(0,0.5,5),rand(12,1));
sol=bvp4c(@dynamic_ric,@bv_ric,sinit,[],A,B,Q,R,x0,P1);
plot(sol.x,sol.y(1:3,:)); figure; plot(sol.x,sol.y(4:end,:));
```

4.5　基于框图的非线性系统的仿真方法

Simulink 环境是 1990 年前后由 The MathWorks 公司推出的产品，原名为 SimuLAB，1992 年改为 Simulink。其名字有两重含义，仿真 (simu) 与模型连接 (link)，表示该环境可以用框图的方式对系统进行仿真。Simulink 提供了各种可用于控制系统仿真的模块，支持一般的控制系统仿真。此外，它还提供了各种工程应用中可能使用的模块，如电机系统、机械系统、通信系统等的模块集，直接进行建模与仿真研究。Simulink 的功能十分强大，可以借用其本身或模块集对任意复杂的系统进行仿真。相关内容可以参阅其手册和书籍[10,11]。

4.5.1　Simulink 简介

在 MATLAB 命令窗口输入 `open_system('simulink')` 命令将打开如图 4-25 所示的模型库，库中还有下一级的模块组，如连续模块组、离散模块组和输入输出模块组等，用户可以用双击的方式打开下一级的模块组，寻找及使用所需要的模块。这里显示的模型库是 Simulink 6.6 版给出的，其他版本的模型库表示形式略有不同。

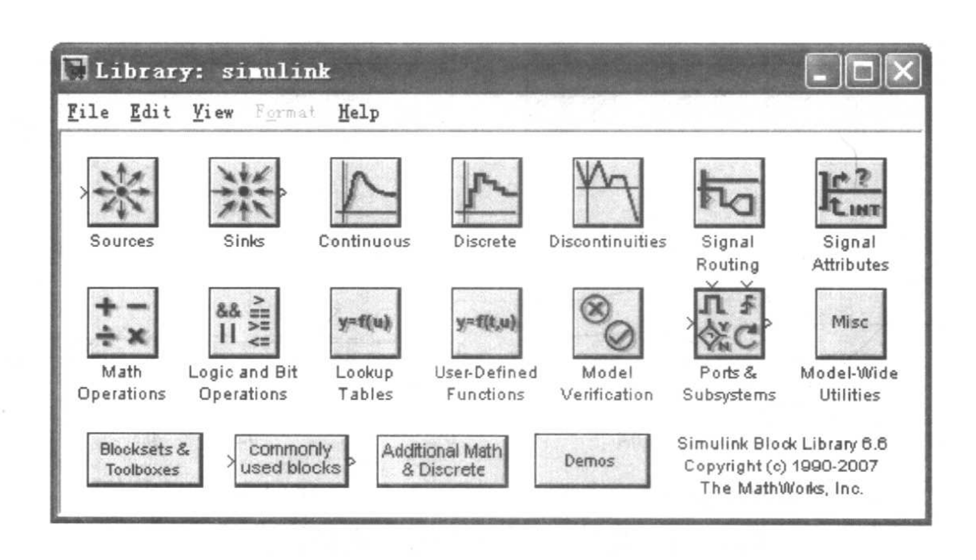


图 4-25　Simulink 主窗口

4.5.2 Simulink 相关模块

Simulink 支持的模块不胜枚举，不可能在本节的篇幅内全部介绍，所以下面将介绍其中常用的模块。

① **输入输出端口** (In1, out1) 一般更常用输出模块显示微分方程求解的结果，该模块将在 MATLAB 工作空间中产生变量 yout。仿真模型中任意一路信号都可以用示波器 Scope 直接显示。

② **时钟模块** Clock 产生时间 t，从而可以搭建时变系统模型。

③ **常用输入模块** 可以用 Sine 模块产生正弦信号，用 Step 可以产生阶跃信号，而用 Constant 模块可以产生恒值信号。

④ **积分器模块** (Int) 可以用其描述一阶导数，令常微分方程组的每个一阶导数项作为每个积分器模块的输入。例如，第 i 个积分器模块的输入端定义为 $\dot{x}_i(t)$，则其输出端自然就是 $x_i(t)$。若给出了一阶微分方程组，则积分器输入端的搭建就是整个微分方程组 Simulink 模型搭建的关键。对于线性高阶微分方程，还可以采用其中的传递函数模块 Transfer Function。

⑤ **延迟模块** (Transport Delay) 可以得出其输入信号在 $t-\tau$ 时刻的值。该模块可以用于延迟微分方程的建模与求解。除了固定时间延迟外，还可以使用 Variable Transport Delay 模块表示可变时间延迟。

⑥ **增益模块** (Gain, Sliding Gain 和 Matrix Gain) 这些增益模块都是建模中很有意义的增益模块，而它们的作用也各不相同。Gain 模块主要用于信号的放大，如果该模块的输入信号为 u，则其输出为 Ku。Matrix Gain 是矩阵增益模块，如果向量型输入信号为 $\boldsymbol{u}$，则其输出信号就成为 $\boldsymbol{K}\boldsymbol{u}$。Sliding Gain 模块较有特色，它实际上是一个滚动杆，用户可以通过鼠标拖动的方式实时改变该模块的增益。

⑦ **数学运算模块** 可以对其输入信号实现加减乘除等代数运算，也可以实现各种逻辑运算和比较运算。

⑧ **数学函数模块** 可以对输入信号做模块指定的非线性运算，如三角函数运算、指数对数运算等。

⑨ **信号向量化模块** 用混路模块 Mux 可以将若干路信号混成向量型的信号，用 DeMux 模块可以将向量型信号解出单路的信号。

4.5.3 基于 Simulink 的控制系统建模与仿真

控制系统仿真研究的一种很常见的要求是通过计算机得出系统在某信号驱动下的时间响应，从中得出期望的结论。如果想研究非线性方程，一方面可以采用前面各节介绍的微分方程数值解法来求解。另一方面，可以采用基于 Simulink 仿

真框图的方法进行求解。因为对于更复杂的系统来说，单纯采用上述的方法有时难以完成仿真任务，比如说，若想研究结构复杂的非线性系统，用前面介绍的方法则需要列写出系统的微分方程，这本身就是很复杂的事，有时甚至是不可能的事。如果有一个基于框图的仿真程序，根据需要可以用框图的形式建立起系统的仿真模型，则解决复杂系统的问题就轻而易举了。这里将通过例子介绍 Simulink 在各种系统中的仿真应用。

1. 混沌问题的仿真建模与研究

混沌方程一般由微分方程描述。仿真这样微分方程有一个技巧，即对每个微分量引入一个积分器，积分器的输出就是该状态变量，那么积分器的输入端就自然是该变量的一阶微分。这样就可以通过搭建仿真模型的方法进行仿真研究。下面通过实际例子演示混沌问题的仿真研究。

例 4-35 再考虑例 4-7 所示的 Rössler 方程，其数学表达式可以重新描述为

$$\begin{cases} \dot{x}(t) = -y(t) - z(t) \\ \dot{y}(t) = x(t) + ay(t) \\ \dot{z}(t) = b + [x(t) - c]z(t) \end{cases}$$

选定 $a = b = 0.2$, $c = 5.7$，且 $x(0) = y(0) = z(0) = 0$。试仿真该系统，绘制出系统的时域响应和相空间轨迹。

求解 对每个微分量应该引入一个积分器，则不难构造如图 4-26 所示的 Simulink 框图，并将三个积分器的初值均设置为 0。在启动仿真过程之前，还可以设置仿真控制参数，如令仿真终止时间为 100，相对误差限为 10^{-7}，这时启动仿真过程，则可以在 MATLAB 工作空间中返回两个变量，tout 和 yout，其中 tout 为列向量，表示各个仿真时刻，而 yout 为一个三列的矩阵，分别对应于三个状态变量 $x_1(t) \sim x_3(t)$。这样用下面的语句就可以绘制出各个状态变量的时间响应曲线，如图 4-27 (a) 所示。

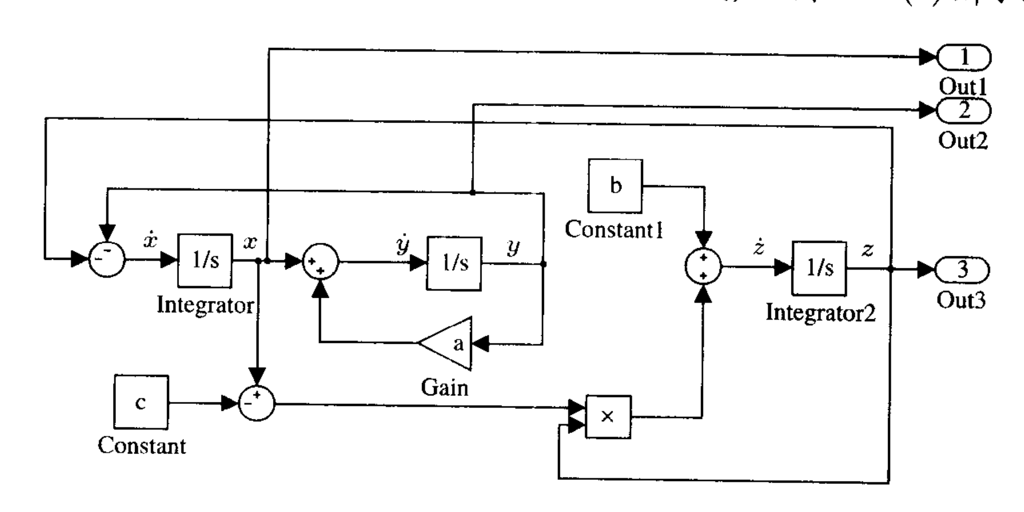
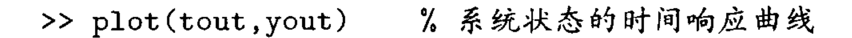
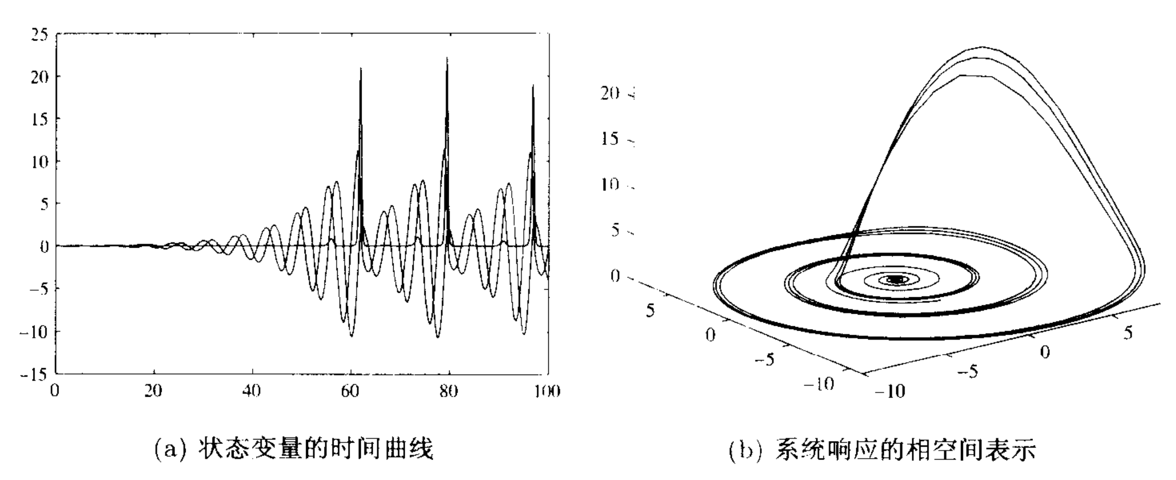


图 4-26 Rössler 方程的 Simulink 表示 (文件名：c4mrossler.mdl)



```
>> plot(tout,yout)    % 系统状态的时间响应曲线
```



(a) 状态变量的时间曲线　　(b) 系统响应的相空间表示

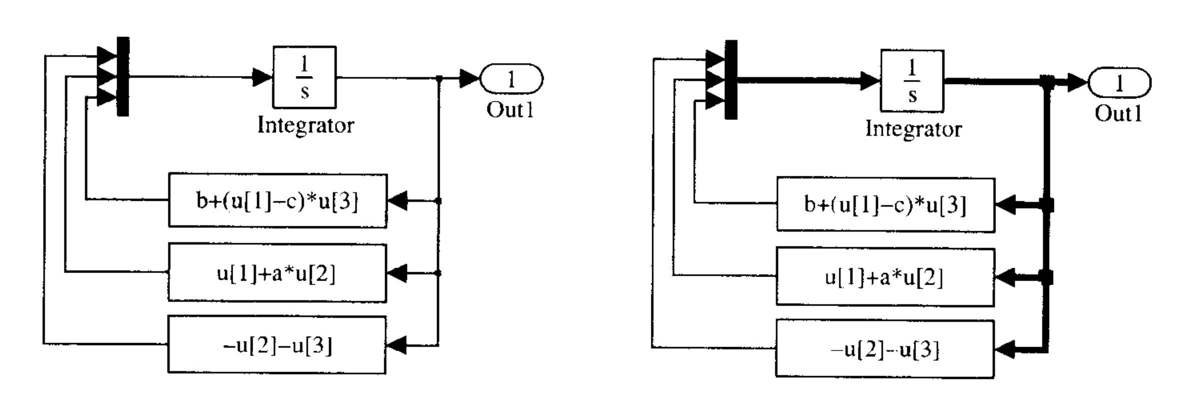
图 4-27　Rössler 方程的仿真结果

若以 $x_1(t)$, $x_2(t)$ 和 $x_3(t)$ 分别为三个坐标轴，这样就可以由下面的语句绘制出三维的相空间曲线，如图 4-27 (b) 所示，comet3() 函数还可以动态演示出状态空间曲线的走向。

```
>> comet3(yout(:,1),yout(:,2),yout(:,3)), grid % 动态显示轨迹
```

注意，这里得出的结果与例 4-7 中的结果完全一致。

Simulink 的模块中很多都支持向量化输入，亦即把若干路信号用 Mux 模块组织成一路信号，这一路信号的各个分量为原来的各路信号。这样这组信号经过积分器模块后，得出的输出仍然为向量化信号，其各路为原来输入信号各路的积分。这样用图 4-28 (a) 中给出的 Simulink 模块就可以改写原来的模型了。在该模型中还使



(a) 改进的仿真框图 (文件名：c4mross1a.mdl)　　(b) 向量型信号线加粗 (文件名：c4mross1b.mdl)

图 4-28　Rössler 方程的其他 Simulink 描述方法

用 Fcn 模块，用于描述对输入信号的数学运算，这里输入信号为系统的状态向量，而 Fcn 模块中将其输入信号记作 u，如果 u 为向量，则用 u[i] 表示其第 i 路分量。若选择 Format→ Port/Signal Display→ Wide Nonscalar Lines 菜单项则会自动加粗向量型信

号，如图 4-28 (b) 所示。可见，这样的系统模型比图 4-26 中给出的 Simulink 模型简洁得多，且这样建模不易出错，也易于维护。

2. 时变系统的 Simulink 仿真

如果系统的某些参数随时间变化，则可以利用 Simulink 输入模块组中的 Clock 模块生成时间变量，用其驱动其他模块搭建时变参数，这种建立起系统的仿真模型。

例 4-36 对时变受控对象模型 $\ddot{y}(t)+\mathrm{e}^{-0.2t}\dot{y}(t)+\mathrm{e}^{-5t}\sin(2t+6)y(t)=u(t)$，考虑一个 PI 控制系统模型，如图 4-29 所示，其中控制器参数为 $K_\mathrm{p}=200$, $K_\mathrm{i}=10$，饱和非线性的宽度为 $\delta=2$，试分析闭环系统的阶跃响应曲线。

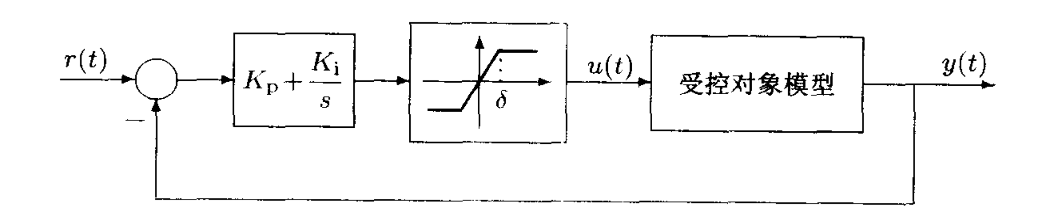


图 4-29 时变控制系统框图

求解 由给出的模型可以看出，除了时变模块外，其他模块的建模是很简单和直观的。对时变部分来说，假设 $x_1(t)=y(t)$, $x_2(t)=\dot{y}(t)$，则可以将微分方程变换成下面的一阶微分方程组

$$\begin{cases} \dot{x}_1(t)=x_2(t) \\ \dot{x}_2(t)=-\mathrm{e}^{-0.2t}x_2(t)-\mathrm{e}^{-5t}\sin(2t+6)x_1(t)+u(t) \end{cases}$$

仿照例 4-35 中使用的方法，给每个状态变量设置一个积分器，则可以搭建起如图 4-30 所示的 Simulink 仿真框图，其中的时变函数用 Simulink 中的函数模块直接表示，注意各个函数模块中函数本身的描述方法是用 u 表示该模块输入信号的，而其输入接时钟模块，生成时变部分的模型，与状态变量用乘法器相乘即可。

建立了仿真模型之后，就可以给出下面的 MATLAB 命令，对该系统进行仿真，并得出该时变系统的阶跃响应曲线，如图 4-31 所示。

```
>> opt=simset('RelTol',1e-8);  % 设置相对允许误差限
   Kp=200; Ki=10;  % 设定控制器参数
   [t,x,y]=sim('c4mtimv',10,opt); plot(t,y)  % 仿真并绘图
```

3. 多变量系统的仿真研究

Simulink 提供的传递函数模块和状态方程模块并不能直接处理带有时间延迟的多变量模型，所以应通过多变量系统的结构，从底层搭建系统的仿真模型。

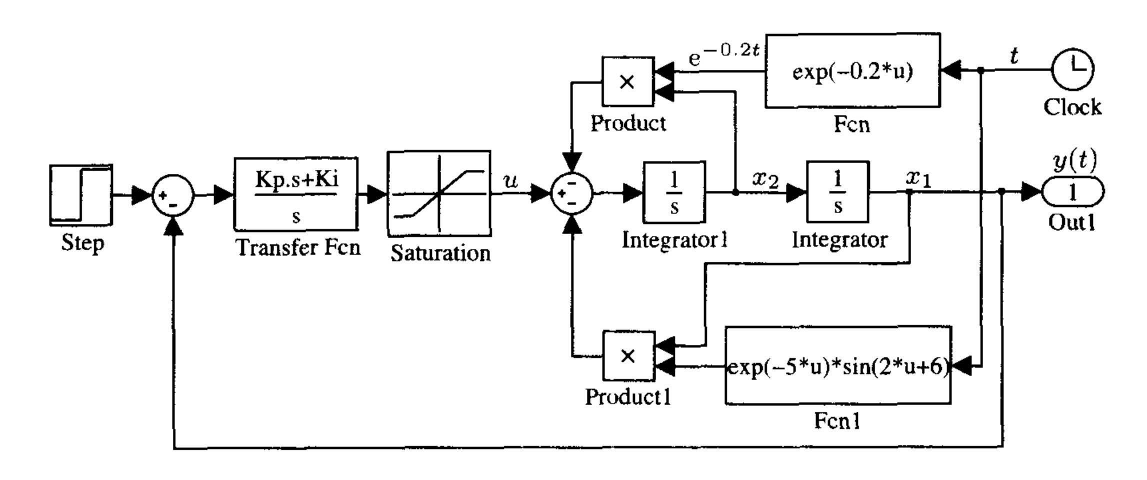


图 4-30 时变系统的 Simulink 表示 (文件名：c4mtimv.mdl)

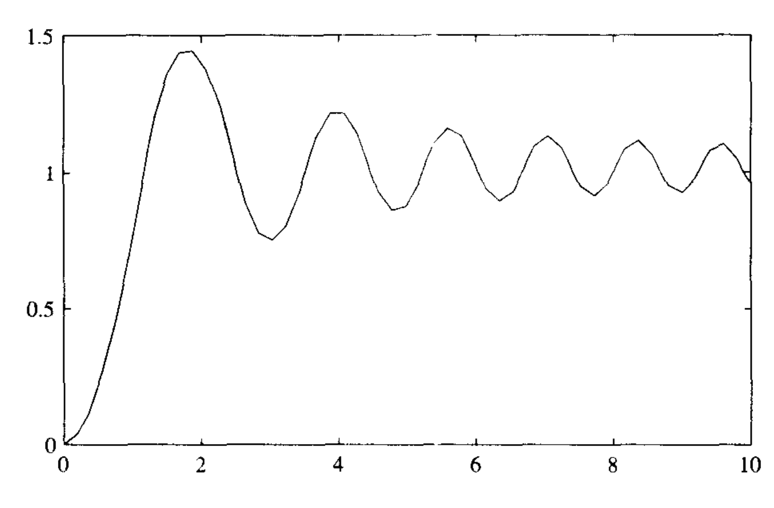


图 4-31 时变系统的阶跃响应曲线

例 4-37 考虑例 2-22 中介绍的多变量系统阶跃响应仿真问题，试构造并仿真该多变量系统模型。

求解 由于含有时间延迟，所以不可能直接用 feedback() 函数构造闭环系统模型，有了 Simulink 这样的工具，就可以容易地建立起精确的仿真模型，如图 4-32 所示。在系统的框图中，分别设置两路阶跃输入的值为 u1 和 u2。

直接用 Simulink 模型进行仿真，则可以容易地得出该系统分别在两路阶跃单独作用下阶跃响应的精确解，并将解析解和近似解在同一坐标系下绘制出来，如图 4-33 所示。

```
>> u1=1; u2=0; [tt1,x1,yy1]=sim('c4mmimo',15); % 第一输入阶跃响应
   u1=0; u2=1; [tt2,x2,yy2]=sim('c4mmimo',15); % 第二输入阶跃响应
   subplot(221), plot(t1,y1(:,1),':',tt1,yy1(:,1))
   subplot(222), plot(t1,y1(:,2),':',tt1,yy1(:,2))
   subplot(223), plot(t2,y2(:,1),':',tt2,yy2(:,1))
```

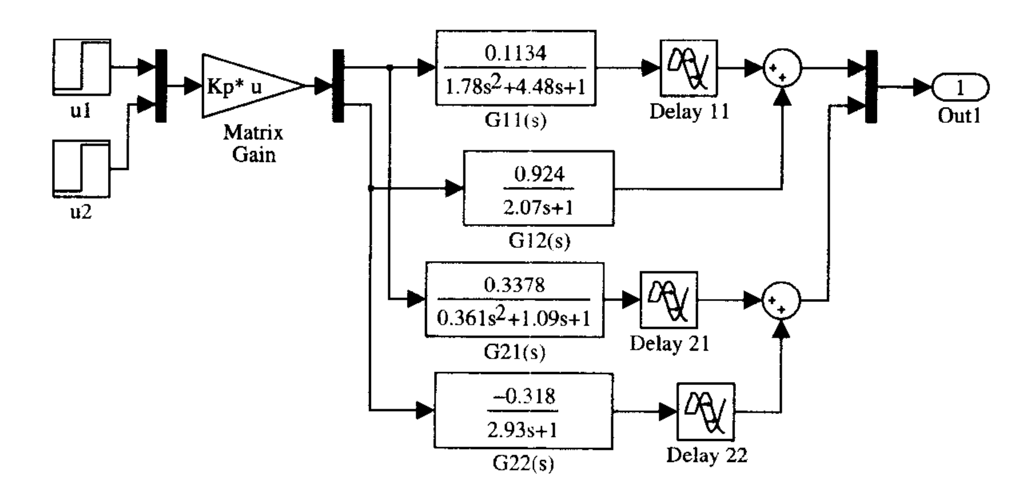


图 4-32　多变量系统的 Simulink 表示 (文件名：c4mmimo.mdl)

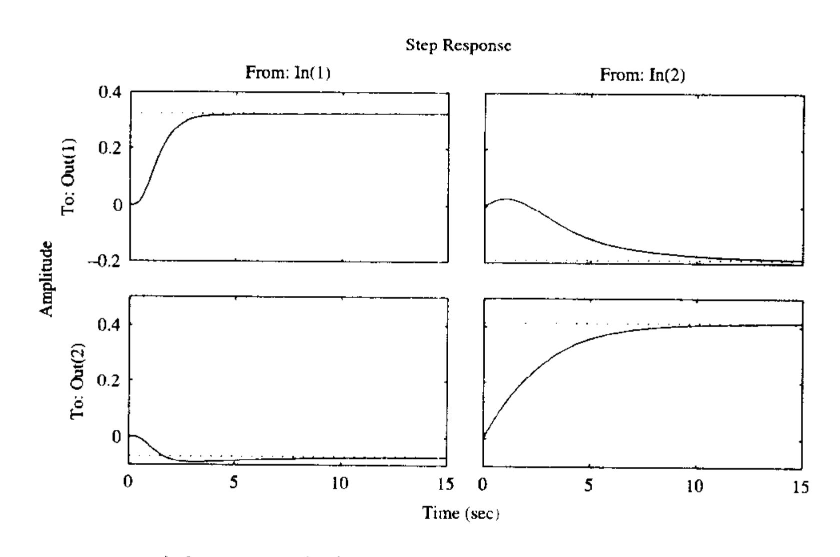


图 4-33　多变量系统的阶跃响应比较

```
subplot(224), plot(t2,y2(:,2),':',tt2,yy2(:,2))
```

在图 4-33 中的实际曲线绘制中，采用实线表示精确仿真结果，虚线表示近似结果，另外对 subplot() 函数设置的坐标系进行了手动修正，使之更加美观。从图中可以看出，这样得出的 Padé 近似结果精度还是比较高的，在得出的图形中几乎看不出二者的差别，所以在一些应用中，可以直接使用 Padé 近似对时间延迟环节进行近似，但应该验证仿真的精度，比如说，看看用不同阶次的 Padé 近似去处理延迟环节，是否能得出相同的仿真结果。

4. 延迟微分方程的求解

如果采用延迟环节，则可以搭建起某信号及其延迟信号，这样采用 Simulink 建模方法，就可以容易地搭建起仿真框图，最终通过仿真的方法得出延迟微分方程的解。

例 4-38 现在考虑例 4-26 中定义的延迟微分方程，其中

$$A_1=\begin{bmatrix}-13 & 3 & -3\\ 106 & -116 & 62\\ 207 & -207 & 113\end{bmatrix},\quad A_2=\begin{bmatrix}0.02 & 0 & 0\\ 0 & 0.03 & 0\\ 0 & 0 & 0.04\end{bmatrix},\quad B=\begin{bmatrix}0\\1\\2\end{bmatrix}$$

试用 Simulink 搭建系统模型，并得出系统的仿真曲线。

求解 该方程用 MATLAB 自身提供的 dde23() 函数无法求解，所以这里考虑采用基于 Simulink 的框图形式求解该方程。在建模之前可以用下面的语句输入已知的矩阵

```
>> A1=[-13,3,-3; 106,-116,62; 207,-207,113];
   A2=diag([0.02,0.03,0.04]); B=[0; 1; 2];
```

再考虑原始的微分方程模型，已经存在一个状态向量 $x(t)$，故可以安排一个积分器，使得其输出为 $x(t)$，这样其输入端自然是 $\dot{x}(t)$，可以分别给这两个信号连接延迟环节，并按实际情况设置延迟时间常数，则可以构造出 $x(t-\tau_1)$ 和 $\dot{x}(t-\tau_2)$ 信号，这样经过简单的处理就可以搭建出如图 4-34 所示的 Simulink 模型。

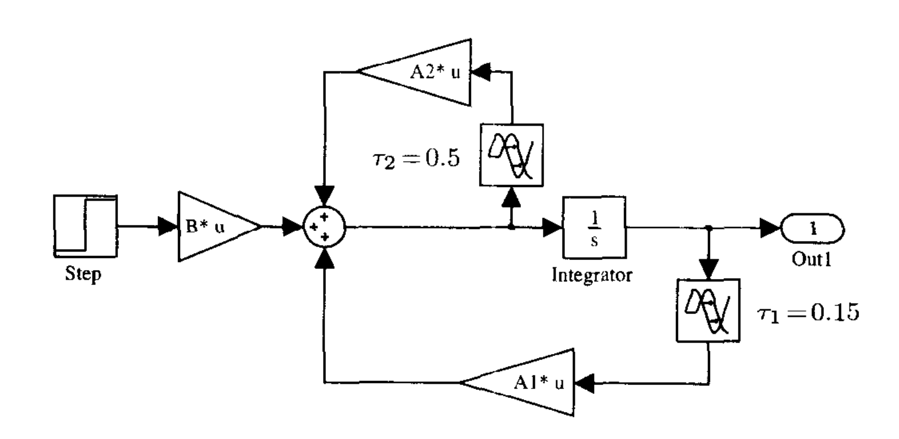


图 4-34 带有导数延迟的微分方程 Simulink 模型 (文件名：c4mdde3.mdl)

用下面的语句就可以求解该方程，并将各个状态变量绘制出来，如图 4-35 所示。可见，用传统求解方法难于求解的中性延迟微分方程也可以容易地求解出来。

```
>> [t,x]=sim('c4mdde3',[0,8]); plot(t,x)
```

5. 网络控制系统的建模与仿真

如果通过计算机网络去对系统进行控制，则系统中存在不确定的时间延迟，这主要表现在系统输出信号检测的延迟和控制信号施加作用的延迟。这种延迟是随机的，此外在网络传输中还可能存在丢包、倒序等现象的出现，这样的现象将影响系统的控制效果。这里将演示带有随机延迟的系统建模与仿真方法。

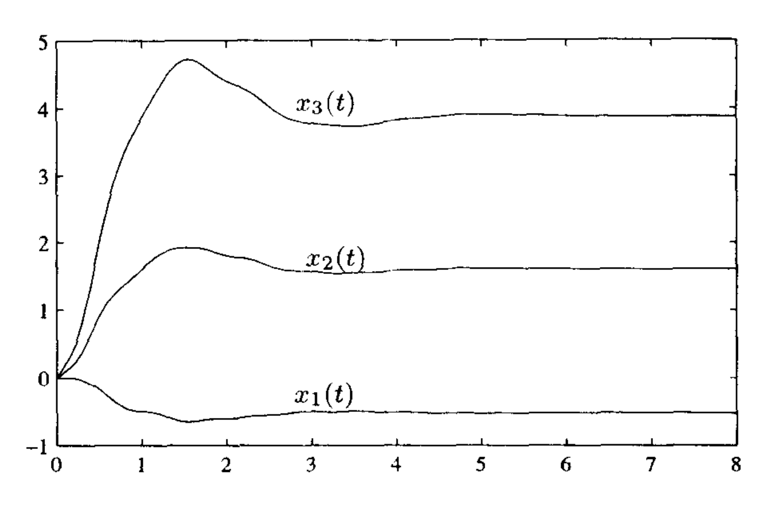


图 4-35 延迟微分方程的数值解

例 4-39 考虑状态方程极点配置算法的网络实现。假设系统的状态方程矩阵为

$$A=\begin{bmatrix}0&2&0&0&-2&0\\1&0&0&0&0&-1\\0&1&0&0&0&0\\0&0&0&3&0&0\\2&0&0&1&0&0\\0&0&-1&0&1&0\end{bmatrix},\quad B=\begin{bmatrix}1&2\\0&0\\0&1\\0&-1\\0&1\\0&0\end{bmatrix}$$

状态变量的初值为 $x_0=[3,4,5,2,3,0]^{\mathrm{T}}$。如果想将系统的闭环极点配置到 $p=-1,-2,-3,-4,-1\pm\mathrm{j}$，试求出状态反馈矩阵。若状态检测和控制信号作用的延迟都是随机的，假设最大的延迟为 0.2 秒，试建立起仿真模型并研究不同的延迟下控制效果。

求解 因为需要所有的状态，所以在输出方程中应该设置 $C=I$。根据要求，由下面的语句即可以设计出状态反馈的矩阵

```
>> A=[0,2,0,0,-2,0; 1,0,0,0,0,-1; 0,1,0,0,0,0; ...
     0,0,0,3,0,0; 2,0,0,1,0,0; 0,0,-1,0,1,0];
   B=[1,2; 0,0; 0,1; 0,-1; 0,1; 0,0]; x0=[3,4,5,2,3,0];
   C=eye(6); D=zeros(6,2); a=0.2;
   p=[-1,-2,-3,-4,-1+1i,-1-1i]; K=place(A,B,p)
```

这样得出的状态反馈矩阵为

$$K=\begin{bmatrix}7.9333&-18.553&-19.134&20.65&18.698&22.126\\-0.36944&-2.0412&-2.3166&-9.5475&0.57469&1.5013\end{bmatrix}$$

现在考虑网络控制的建模。引入随机延迟模块，则可以构造出如图 4-36 所示的仿真模型，其中 a 为两个延迟模块的最大延迟值。仿真模型中未考虑通信丢包现象。

设置 $a=10^{-5}$，则可以认为检测和控制没有延迟，这时可以得出直接状态反馈控制的控制效果，如图 4-37 (a) 所示。令 $a=0.2$，则得出的网络控制效果如图 4-37 (b)

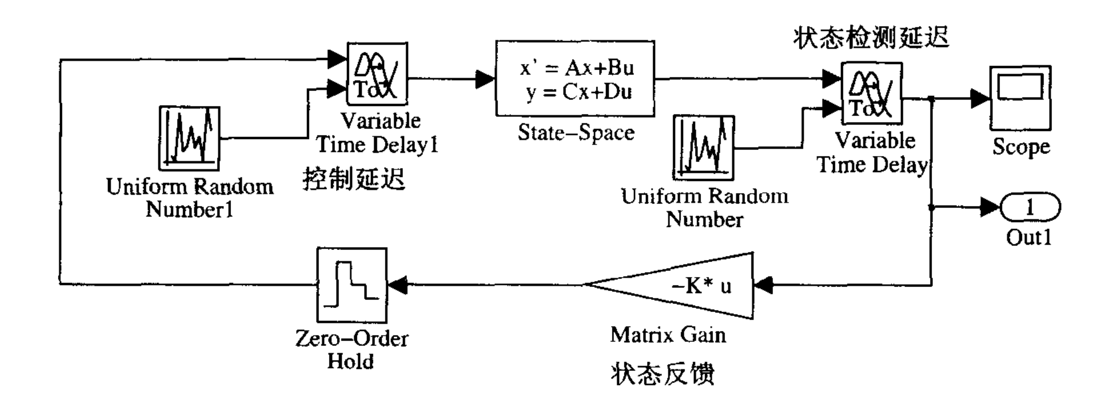
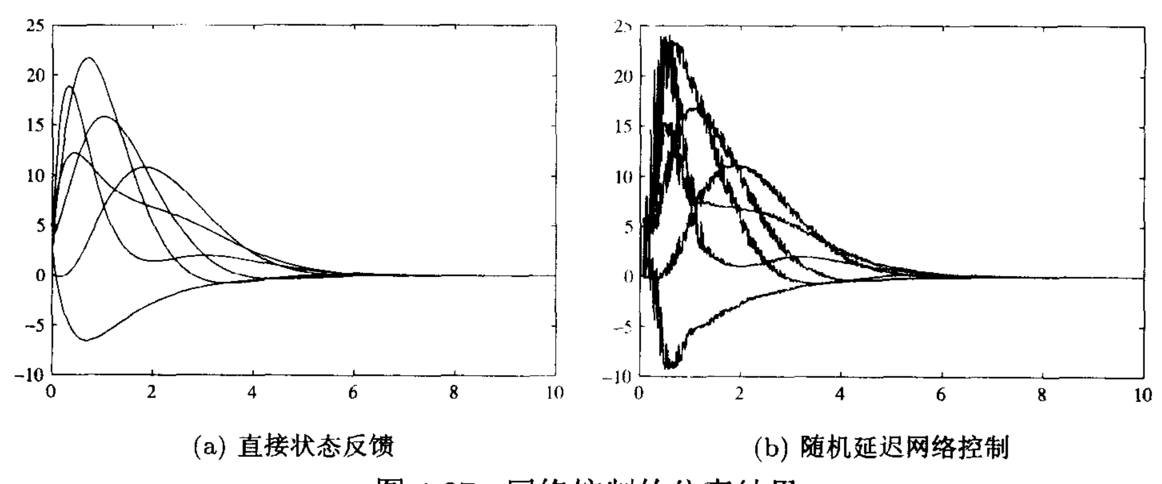


图 4-36 网络状态反馈极点配置控制的仿真框图 (文件名：c4mnet.mdl)

所示。可见，控制效果比较理想。



(a) 直接状态反馈

(b) 随机延迟网络控制

图 4-37 网络控制的仿真结果

进一步增大 a 值，则直至 $a = 0.4$，网络控制系统仍然稳定，但网络控制效果明显变差。若进一步增大延迟，如 $a = 0.5$，则系统不稳定。

6. 一般非线性模块的搭建

Simulink 库中提供了一些分段线性的非线性静态模块，如饱和非线性模块、死区非线性模块等。除了这些模块外，还可以用查表模块搭建任意的单值和单值非线性模块[12]。单值非线性环节可以通过查表模块直接搭建。双值非线性搭建麻烦一些，搭建方法可以由下面的例子演示。

例 4-40 试用 Simulink 模块搭建起如图 4-38 (a) 所示的非线性系统模型，积分器的初始状态 $x_1(0) = 1$，其中的非线性环节如图 4-38 (b) 所示的双值非线性模块。试对该系统进行仿真研究。

求解 分析给出的系统框图可见，本问题的难点是双值非线性回环环节的构造。该回环在输入量减小时由图 4-39 (a) 中给出的下降分支计算输出量，而增加时由图 4-39 (b) 中的上升分支计算。

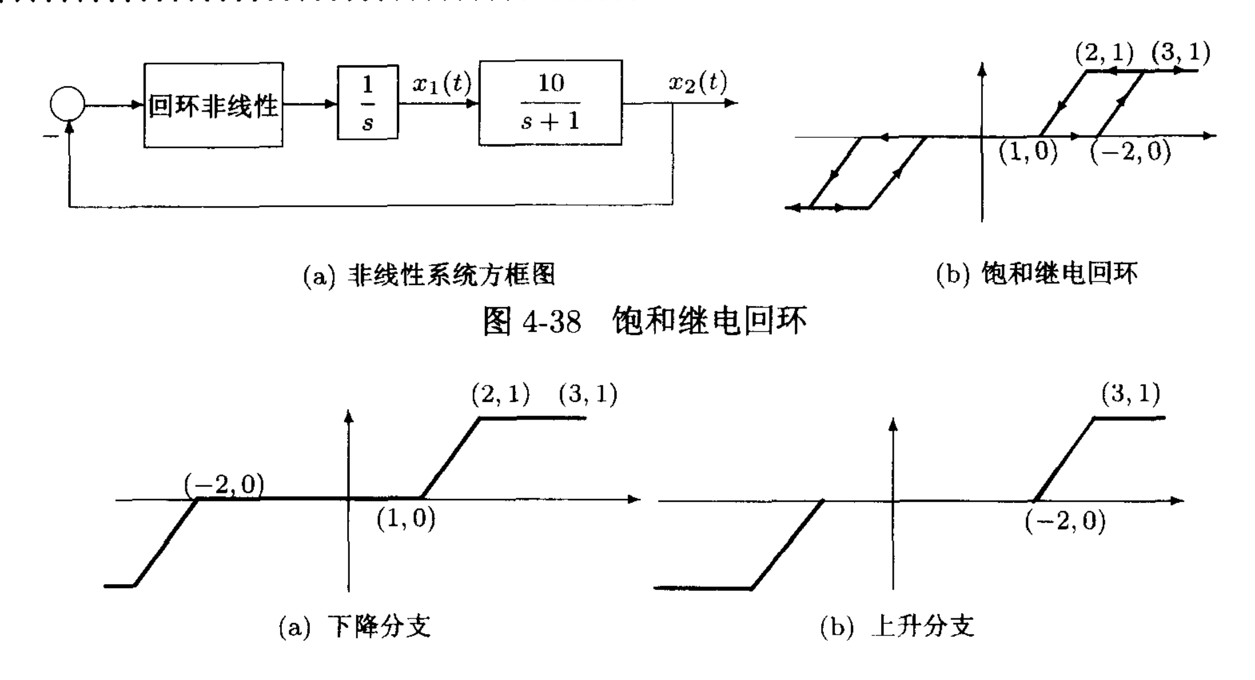


图 4-38　饱和继电回环

图 4-39　分解成两个单值非线性环节

Simulink 的连续模块组中提供了一个 Memory (记忆) 模块，该模块记忆上一个计算步长上的信号值，所以可以按照图 4-40 中所示的格式构造一个 Simulink 模型。在该框图中使用了一个比较符号来比较当前的输入信号与上一步输入信号的大小，其输出是逻辑变量，在上升时输出的值为 1，下降时的值为 0。由该信号可以控制后面的开关模块，设开关模块的阈值 (Threshold) 为 0.5，则当输入信号为上升时由上面的通路计算整个系统的输出，而下降时由下面的通路计算输出。

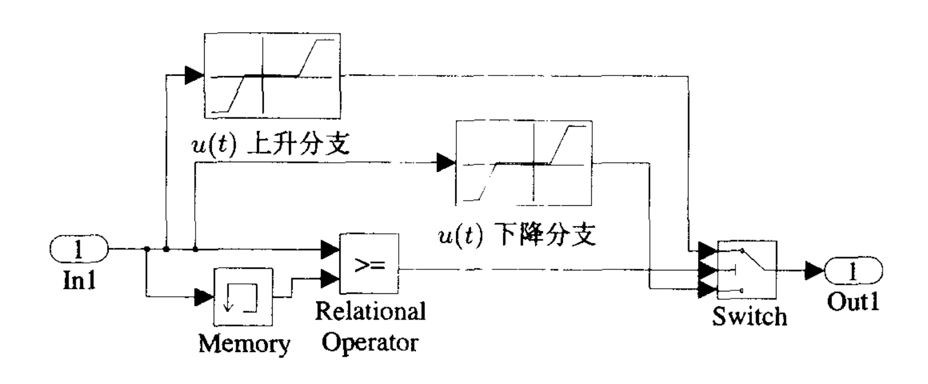


图 4-40　非线性模块的 Simulink 表示 (文件名：c4mloopa.mdl)

7. 连续系统随机输入仿真研究

连续系统在随机噪声驱动下，其系统内信号不能简单地用随机数作为输入进行仿真，而必须采用和仿真步长相关的加权随机信号，已保证在一个指定的频率范围内保持一个恒定的输入功率，最理想的近似输入信号为带宽受限的输入信号[13]。为了得出系统的近似解，必须使用定步长的仿真方法，并对伪随机输入函

数进行比例化。假定比例化系数为 K_σ，即 $\gamma_i = K_\sigma e_i$，式中 e_i 为满足正态分布的伪随机数，而 γ_i 为可以用于仿真的实际输入量。假定在一个计算步长内输入信号 γ_i 为一常值，即

$$\gamma(t) = \gamma_i = K_\sigma e_i,\ \ i\Delta t \leqslant t < (i+1)\Delta t \tag{4-5-1}$$

为保证伪随机变量 γ_i 的强度与原随机信号 $\gamma_0(t)$ 相同，则可以证明[13] $K_\sigma = \sigma/\sqrt{\Delta t}$。可见，如果单纯采用随机数作为输入将得出错误的仿真结果。在 Simulink 环境中，采用 Band-Limited White Noise 模块作为输入源即可以模拟白噪声输入。

例 4-41 假设已知非线性系统的框图如图 4-41 所示，其中线性部分和死区非线性部分分别为[14]

$$G(s) = \frac{14s^3+248s^2+900s+1200}{s^4+18s^3+102s^2+180s+120},\ \ N(e) = \begin{cases} me - m\delta\mathrm{sgn}(e), & |e| > \delta \\ 0, & |e| \leqslant \delta \end{cases}$$

输入 $r(t)=0$，扰动 $\gamma(t)$ 为白噪声信号，且非线性特性 $m=2, \delta=1$，试对该系统进行仿真，并得出误差信号 $e(t)$ 的概率密度。

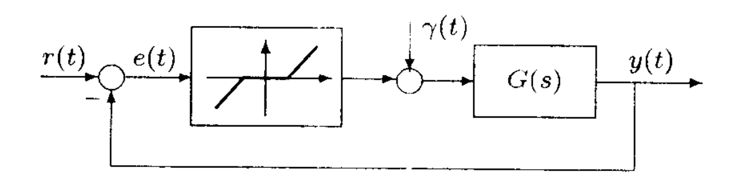


图 4-41 非线性系统框图及仿真模型

求解 根据需要可以构造出如图 4-42 所示的仿真模型。设置扰动信号 $\gamma(t)$ 的采样周

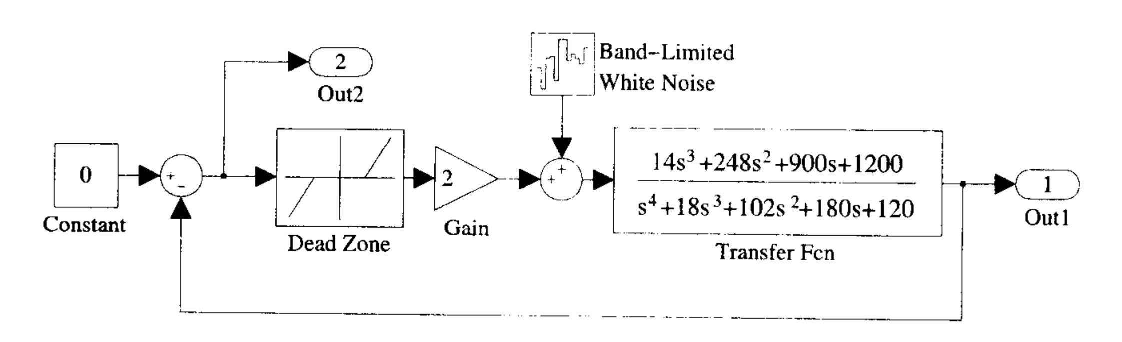
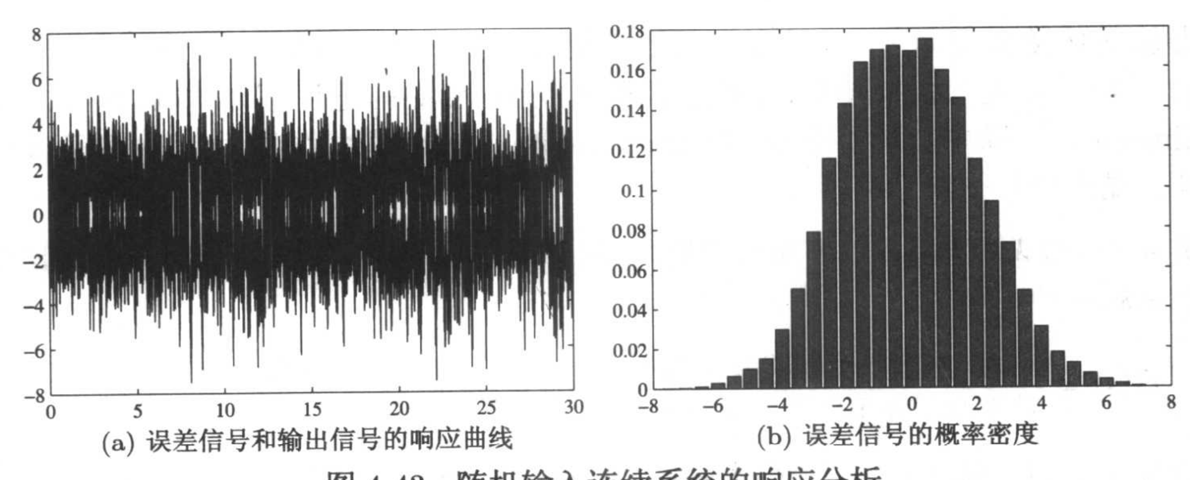


图 4-42 非线性系统仿真模型 (c4mrnd.mdl)

期为 0.001，计算 30000 个点。并选择步长为 0.001 的定步长仿真算法，对该系统进行仿真，则可以得出相应的输出曲线和误差曲线，如图 4-43 (a) 所示。显然，这样的输出信号杂乱无章，所以应该侧重于研究误差信号的统计特性。由下面语句可以得出误

差信号的概率密度曲线，如图 4-43 (b) 所示。非线性系统信号的理论解可以由相应的 Fokker-Planck 方程得出 [14]。

```
>> plot(tout,yout), figure  % 绘制系统的响应曲线
   x=-8:0.5:8; N=hist(yout(:,2),x); y=N/0.5/30000; bar(x,y)
```



(a) 误差信号和输出信号的响应曲线　(b) 误差信号的概率密度

图 4-43　随机输入连续系统的响应分析

4.5.4　S-函数的设计与应用

如果某些系统不利于底层模块搭建，则可以采用 S-函数的形式来描述，S 是系统的意思，而 S-函数实际上就是状态方程的语句描述。

S-函数是有固定格式的，MATLAB 语言和 C 语言编写的 S-函数的格式是不同的。用 MATLAB 语言编写的 S-函数的引导语句为：

function [sys,x_0,str,ts]=fun(t,x,u,flag,p_1,p_2,$\cdots$)

其中 `fun` 为 S-函数的函数名，t, x, u 分别为时间、状态和输入信号，`flag` 为标志位，标志位的取值不同，S-函数执行的任务与返回数据也是不同的。

① 若 `flag` 的值为 0 时，将启动 S-函数所描述系统的初始化过程，这时将调用一个名为 `mdlInitializeSizes()` 的子函数，该函数应该对一些参数进行初始设置，如离散状态变量的个数、连续状态变量的个数，模块输入和输出的路数，模块的采样周期个数和采样周期的值、模块状态变量的初值向量 x_0 等。首先通过 `sizes=simsizes` 语句获得默认的系统参数变量 `sizes`。得出的 `sizes` 实际上是一个结构体变量，其常用成员为：

`.NumContStates` 表示 S-函数描述的模块中连续状态的个数。

`.NumDiscStates` 表示离散状态的个数。

`.NumInputs` 和 `NumOutputs` 分别表示模块输入和输出的个数。

`.DirFeedthrough` 为输入信号是否直接在输出端出现，取值可以为 0, 1。

.NumSampleTimes 为模块采样周期的个数，S-函数支持多采样周期系统。

按照要求设置好的结构体 sizes 应该在通过 sys=simsizes(sizes) 语句赋给 sys 参数。除了 sys 外，还应该设置系统的初始状态变量 x_0、说明变量 str 和采样周期变量 ts，其中 ts 变量应该为双列的矩阵，其中每一行对应一个采样周期。对连续系统和有单个采样周期的系统来说，该变量为 $[t_1, t_2]$，其中 t_1 为采样周期，如果取 $t_1 = -1$ 则将继承输入信号的采样周期参数 t_2 为偏移量，一般取为 0。

② 若 flag 的值为 1 时，将作连续状态变量的更新，即调用 mdlDerivatives() 函数，更新后的连续状态变量将由 sys 变量返回。

③ 若 flag 的值为 2 时，将作离散状态变量的更新，即调用 mdlUpdate() 函数，更新后的离散状态变量将由 sys 变量返回。

④ 若 flag 的值为 3 时，将求取系统的输出信号，即调用 mdlOutputs() 函数，将计算得出的输出信号由 sys 变量返回。

⑤ 若 flag 的值为 4 时，将调用 mdlGetTimeOfNextVarHit() 函数，计算下一步的仿真时刻，并将计算得出的下一步仿真时间由 sys 变量返回。

⑥ 若 flag 的值为 9 时，将终止仿真过程，即调用 mdlTerminate() 函数，这时不返回任何变量。

S-函数中目前不支持其他的 flag 选择。形成 S-函数的模块后，就可以将其嵌入到系统的仿真模型中进行仿真了。在实际仿真过程中，Simulink 会自动将 flag 设置成 0，进行初始化过程，然后将 flag 的值设置为 3，计算该模块的输出。再将 flag 的值分别设置为 1 和 2，更新系统的连续和离散状态。这样就可以完成一个仿真周期的计算。一个仿真周期后，在将其顺序地设置为 1,2,3，如此一个周期接一个周期地计算，直至仿真结束条件满足，Simulink 自动将把 flag 的值设置成 9，终止仿真过程。

S-函数编写有几个部分应该注意，首先是初始化编程，程序设计者应该首先弄清楚系统的输入、输出信号是什么，模块中应该有多少个连续状态，多少个离散状态，离散模块的采样周期是什么等基本信息，有了这些信息就可以进行模块的初始化了。初始化过程结束后，还应该知道该模块连续和离散的状态方程分别是什么，如何用 MATLAB 语句将其表示出来，并应该清楚如何从模块的状态和输入信号计算模块的输出信号，这样就可以编写系统的状态方程、离散状态更新及模块的输出计算部分，从而完成 S-函数的编写了。这里将通过一些例子介绍 S-函数的编写方法。

例 4-42 这里通过微分–跟踪器介绍 S-函数的编写，微分–跟踪器[15]的离散形式为

$$\begin{cases} x_1(k+1) = x_1(k) + Tx_2(k) \\ x_2(k+1) = x_2(k) + T\mathrm{fst}(x_1(k), x_2(k), u(k), r, h) \end{cases} \tag{4-5-2}$$

式中，T 为采样周期，$u(k)$ 为第 k 时刻的输入信号，r 为决定跟踪快慢的参数，而 h 为输入信号被噪声污染时，决定滤波效果的参数。fst 函数可以由下面的式子计算

$$\delta = rh,\ \ \delta_0 = \delta h,\ \ y = x_1 - u + hx_2,\ \ a_0 = \sqrt{\delta^2 + 8r|y|} \tag{4-5-3}$$

$$a = \begin{cases} x_2 + y/h, & |y| \leqslant \delta_0 \\ x_2 + 0.5(a_0 - \delta)\mathrm{sgn}(y), & |y| > \delta_0 \end{cases} \tag{4-5-4}$$

$$\mathrm{fst} = \begin{cases} -ra/\delta, & |a| \leqslant \delta \\ -r\mathrm{sgn}(a), & |a| > \delta \end{cases} \tag{4-5-5}$$

试用 S-函数编写出通用的微分-跟踪器模型。

求解　可以看出，该算法直接用 Simulink 模块搭建还是比较困难的，所以这里将介绍采用 S-函数建立该模块的方法。从式 (4-5-2) 中给出的状态方程可以看出，系统有两个离散状态，$x_1(k)$ 和 $x_2(k)$，没有连续状态，有一路输入信号 $u(k)$，另外微分-跟踪器应该输出两路信号，原输入信号的跟踪信号 $y_1(k) = x_1(k)$ 和其微分 $y_2(k) = x_2(k)$，系统的采样周期为 T，由于系统的输出可以由状态直接计算出，不直接涉及输入信号 $u(k)$，所以初始化中 DirectFeedthrough 属性应该设置为 0。另外，r, h, T 还应该理解成该模块的附加参数。根据上述算法，立即可以写出其相应的 S-函数实现。

```
function [sys,x0,str,ts]=han_td(t,x,u,flag,r,h,T)
switch flag,
case 0 % 调用初始化函数
   [sys,x0,str,ts] = mdlInitializeSizes(T);
case 2 % 调用离散状态的更新函数
   sys = mdlUpdates(x,u,r,h,T);
case 3, sys=x; % 直接计算输出量
case {1, 4, 9} % 未使用的 flag 值
   sys = [];
otherwise % 处理错误
   error(['Unhandled flag = ',num2str(flag)]);
end;
% 当 flag 为 0 时进行整个系统的初始化
function [sys,x0,str,ts] = mdlInitializeSizes(T)
% 首先调用 simsizes 函数得出系统规模参数 sizes，并根据离散系统
% 的实际情况设置 sizes 变量
sizes = simsizes;     % 读入初始化参数模板
sizes.NumContStates = 0;    % 无连续状态
sizes.NumDiscStates = 2;    % 有两个离散状态
sizes.NumOutputs = 2;% 输出两个量：跟踪信号和微分信号
sizes.NumInputs = 1; % 系统输入信号一路
sizes.DirFeedthrough = 0;   % 输入不直接传到输出口
```

```
sizes.NumSampleTimes = 1;  % 单个采样周期
sys = simsizes(sizes);     % 根据上面的设置设定系统初始化参数
x0 = [0; 0];   % 设置初始状态为零状态
str = [];% 将 str 变量设置为空字符串即可
ts = [T 0];    % 采样周期，若写成 -1 则表示继承其输入信号
% 在主函数的 flag=2 时，更新离散系统的状态变量
function sys = mdlUpdates(x,u,r,h,T)
sys=[x(1)+T*x(2); x(2)+T*fst2(x,u,r,h)];
% 用户定义的子函数: fst2
function f=fst2(x,u,r,h)
delta=r*h; delta0=delta*h; y=x(1)-u+h*x(2);
a0=sqrt(delta*delta+8*r*abs(y));
if abs(y)<=delta0, a=x(2)+y/h;
else, a=x(2)+0.5*(a0-delta)*sign(y); end
if abs(a)<=delta, f=-r*a/delta; else, f=-r*sign(a); end
```

编写了 S-函数模块后，就可以在仿真模型中利用该模块了。例如在图 4-44 中给出的仿真框图中，直接使用了编写的 S-函数模块 han_td，其输入端为信号发生器模块，输出端直接接示波器。双击其中的 S-函数模块，则将打开如图 4-45 所示的参数对话框，允许用户输入 S-函数的附加参数。在对话框中，输入 $r = 30$, $h = 0.01$ 与 $T = 0.001$，并令输入信号为正弦信号，并选择仿真算法为定步长，步长为 0.001，则可以对系统进行仿真分析，得出如图 4-46 所示的仿真结果。

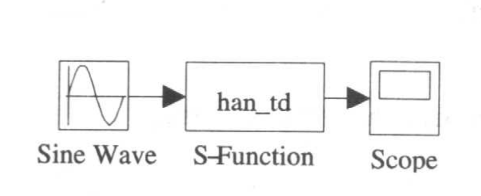


图 4-44 仿真模型 (c4msf2.mdl)

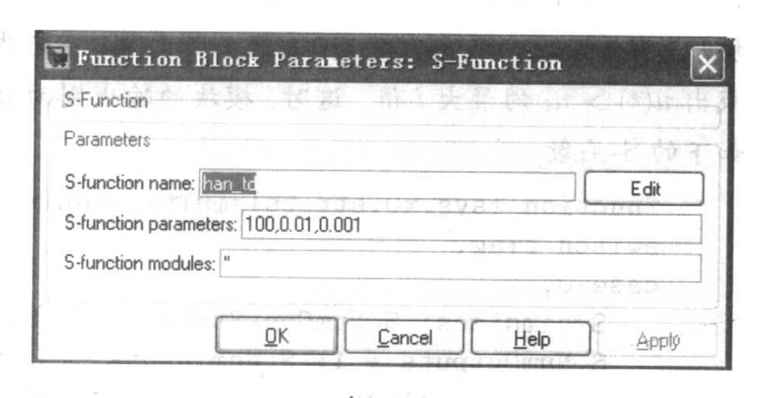


图 4-45 S-函数参数设置对话框

例 4-43 在很多控制系统仿真研究领域，通常需要一种多阶梯函数作为驱动信号。假设想在 $t_1, t_2, \cdots, t_N$ 时刻分别开始生成幅值为 $r_1, r_2, \cdots, r_N$ 的阶跃信号，如图 4-47 所示，这样的模块用 Simulink 现有的模块搭建是很麻烦的，如果 N 很大，则特别难以实现。试用 S-函数编写这样的模块并用封装技术构造相应的多阶梯函数模块。

求解 显然，若想用 S-函数描述次模块，则需要将 $\boldsymbol{t} = [t_1, \cdots, t_N]$ 和 $\boldsymbol{u} = [r_1, \cdots, r_N]$ 作为附加变量输入给模块。这样系统的有一路输入信号 $u(t)$，有一路输出信号 $y(t)$。

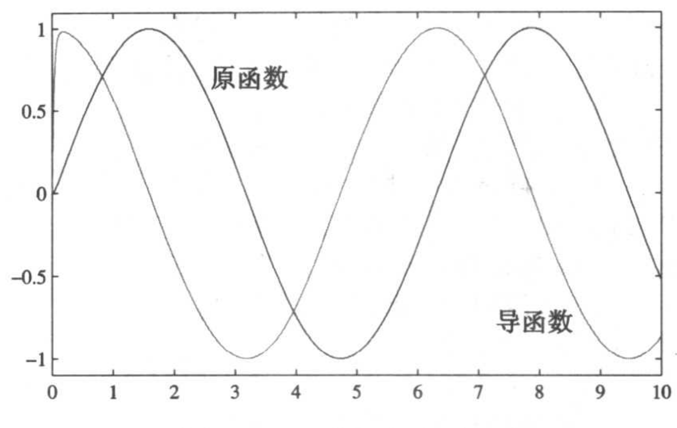
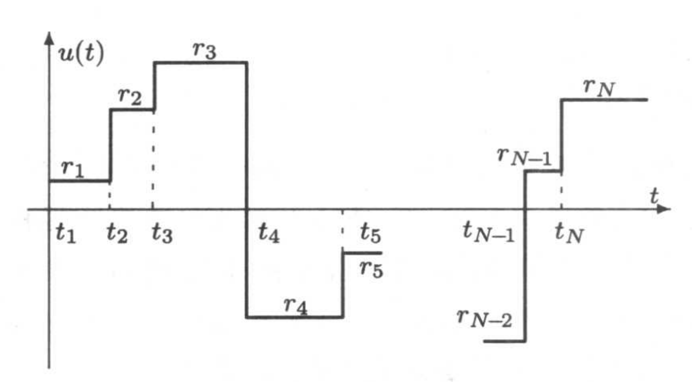


图 4-46　系统仿真结果



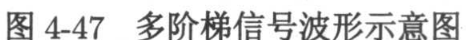


图 4-47　多阶梯信号波形示意图

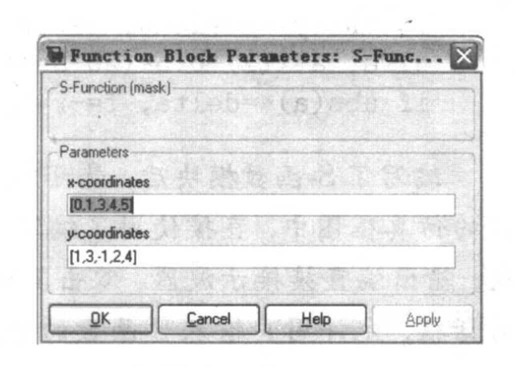


图 4-48　多阶梯输入模块封装

由于可以由 $u(t)$ 和 $\boldsymbol{t}$, $\boldsymbol{u}$ 向量直接计算出输出信号 $y(t)$，所以此模块应该是纯静态模块，没有连续状态和离散状态。由已知 $\boldsymbol{t}$ 向量，首先找出当前的 $u(t)$ 属于哪个段落，即找出 $u(t) > u_i$ 的最大 i 值，这时，模块的输出则为 $y(t) = r_i$。由这个算法可以编写出如下的 S-函数

```
function [sys,x0,str,ts]=multi_step(t,x,u,flag,tTime,yStep)
switch flag,
case 0,
   S=simsizes; S.NumContStates = 0; S.NumDiscStates = 0;
   S.NumOutputs = 1; S.NumInputs = 0; S.DirFeedthrough = 0;
   S.NumSampleTimes=1; sys=simsizes(S); x0=[]; str=[]; ts=[0 0];
case 3,
   i=find(tTime<=t); sys=yStep(i(end));
case {1, 2, 4, 9},  sys = [];
otherwise, error(['Unhandled flag = ',num2str(flag)]);
end;
```

如果想构造一个单独的多阶梯函数模块，则可以将一个 S-函数模块复制到空白的 Simulink 模型窗口，双击该模块，在对话框中将其和 `multi_step.m` 建立联系，在附加参数栏目填写 `tvec,uvec`。右击该模块，选中菜单中的 Mask S-Function 项，则可以

得出封装模块对话框，在其 Parameters 栏目下定义两个对话框输入变量 tvec,uvec，在 Initialization 栏目给出下面的命令准备模块绘图数据

```
x=sort([tvec,tvec(2:end)-1e-5,2*tvec(end)-tvec(end-1)]);
y0=[uvec(:) uvec(:)]'; y=y0(:);
```

这样在 Icon 栏目给出命令 plot(x,y) 即可以给该图标绘制多阶梯函数的波形。双击该封装模块，则可以得出如图 4-48 所示的对话框，用户可以在该对话框中提供 t 和 u 向量来使用它。为后面内容使用，可以将其存成 c4mmstep.mdl 文件。

4.6 习题与思考题

1 试求出下面线性微分方程的通解。

$$\frac{\mathrm{d}^5y(t)}{\mathrm{d}t^5}+13\frac{\mathrm{d}^4y(t)}{\mathrm{d}t^4}+64\frac{\mathrm{d}^3y(t)}{\mathrm{d}t^3}+152\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+176\frac{\mathrm{d}y(t)}{\mathrm{d}t}+80y(t)$$
$$=\mathrm{e}^{-2t}\left[\sin\left(2t+\frac{\pi}{3}\right)+\cos(3t)\right]$$

假设上述微分方程满足已知条件 $y(0)=1, y(1)=3, y(\pi)=2, \dot{y}(0)=1, \dot{y}(1)=2$，试求出满足该条件的微分方程的解析解。

2 试求解下面微分方程的通解以及满足 $x(0)=1, x(\pi)=2, y(0)=0$ 条件下的解析解。

$$\begin{cases}\ddot{x}(t)+5\dot{x}(t)+4x(t)+3y(t)=\mathrm{e}^{-6t}\sin(4t)\\ 2\dot{y}(t)+y(t)+4\dot{x}(t)+6x(t)=\mathrm{e}^{-6t}\cos(4t)\end{cases}$$

3 试求出下面的时变线性微分方程解析解。

① Legendre 微分方程 $(1-t^2)\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}-2t\dfrac{\mathrm{d}x}{\mathrm{d}t}+n(n+1)x=0$

② Bessel 微分方程 $t^2\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}+t\dfrac{\mathrm{d}x}{\mathrm{d}t}+(t^2-n^2)x=0$

4 试求出微分方程 $\ddot{y}(x)-\left(2-\dfrac{1}{x}\right)\dot{y}(x)+\left(1-\dfrac{1}{x}\right)y(x)=x^2\mathrm{e}^{-5x}$ 的解析解通解，并求出满足边界条件 $y(1)=\pi$, $y(\pi)=1$ 的解析解。

5 试用 Laplace 变换求解下面微分方程组的解析解，并和其他方法比较。

$$\begin{cases}\ddot{x}(t)+\ddot{y}(t)+x(t)+y(t)=0,\ x(0)=2, y(0)=1\\ 2\ddot{x}(t)-\ddot{y}(t)-x(t)+y(t)=\sin t,\ \dot{x}(0)=\dot{y}(0)=-1\end{cases}$$

6 试求出下面微分方程的通解。

① $\ddot{x}(t)+2t\dot{x}(t)+t^2x(t)=t+1$ ② $\dot{y}(x)+2xy(x)=x\mathrm{e}^{-x^2}$

③ $y^{(3)}+3\ddot{y}+3\dot{y}+y=\mathrm{e}^{-t}\sin t$

7 极限环是二阶非线性常微分方程中一种常见的现象，对某些非线性微分方程来说，不论初始状态为何值，微分方程的相轨迹都将稳定在一条封闭的曲线上，该曲线称为微分方程的极限环。试绘制出微分方程 $\begin{cases} \dot{x} = y + x(1 - x^2 - y^2) \\ \dot{y} = -x + y(1 - x^2 - y^2) \end{cases}$ 的极限环，并对不同初值验证微分方程的相平面曲线确实收敛于极限环。

8 Chua 电路方程是混沌理论中经常提到的微分方程[16] $\begin{cases} \dot{x} = \alpha[y - x - f(x)] \\ \dot{y} = x - y + z \\ \dot{z} = -\beta y - \gamma z \end{cases}$，其中，$f(x)$ 为 Chua 电路的二极管分段线性特性

$$f(x) = bx + \frac{1}{2}(a - b)(|x + 1| - |x - 1|), \text{且 } a < b < 0$$

试编写出 MATLAB 函数描述该微分方程，并绘制出 $\alpha = 15, \beta = 20, \gamma = 0.5, a = -120/7, b = -75/7$，且初始条件为 $x(0) = -2.121304, y(0) = -0.066170, z(0) = 2.881090$ 时的相空间曲线。

9 Lotka-Volterra 捕食模型方程为 $\begin{cases} \dot{x}(t) = 4x(t) - 2x(t)y(t) \\ \dot{y}(t) = x(t)y(t) - 3y(t) \end{cases}$，且初值为 $x(0) = 2, y(0) = 3$，试求解该微分方程，并绘制相应的曲线。

10 请给出求解下面微分方程的 MATLAB 命令，

$$y^{(3)} + ty\ddot{y} + t^2\dot{y}y^2 = \mathrm{e}^{-ty}, \ y(0) = 2, \ \dot{y}(0) = \ddot{y}(0) = 0$$

并绘制出 $y(t)$ 曲线。试问该方程存在解析解吗？选择四阶定步长 Runge-Kutta 算法求解该方程时，步长选择多少可以得出较好的精度，试与 MATLAB 语言给出的现成函数在速度、精度上进行比较。

11 试选择状态变量，将下面的非线性微分方程组转换成一阶显式微分方程组，并用 MATLAB 对其求解，绘制出解的相平面或相空间曲线。

① $\begin{cases} \ddot{x} = -x - y - (3\dot{x})^2 + (\dot{y})^3 + 6\ddot{y} + 2t \\ y^{(3)} = -\ddot{y} - \dot{x} - \mathrm{e}^{-x} - t \\ x(1) = 2, \dot{x}(1) = -4 \\ y(1) = -2, \dot{y}(1) = 7, \ddot{y}(1) = 6 \end{cases}$

② $\begin{cases} \ddot{x} - 2xz\dot{x} = 3x^2yt^2 \\ \ddot{y} - \mathrm{e}^y\dot{y} = 4xt^2z \\ \ddot{z} - 2t\dot{z} = 2t\mathrm{e}^{xy} \\ \dot{z}(1) = \dot{x}(1) = \dot{y}(1) = 2 \\ \dot{z}(1) = x(1) = y(1) = 3 \end{cases}$

12 试用解析解和数值解的方法求解下面的微分方程组。

$$\begin{cases} \ddot{x}(t) = -2x(t) - 3\dot{x}(t) + \mathrm{e}^{-5t}, & x(0) = 1, \dot{x}(0) = 2 \\ \ddot{y}(t) = 2x(t) - 3y(t) - 4\dot{x}(t) - 4\dot{y}(t) - \sin t, & y(0) = 3, \dot{y}(0) = 4 \end{cases}$$

13 给定微分方程组 $\begin{cases} \ddot{u}(t) = -u(t)/r^3(t) \\ \ddot{v}(t) = -v(t)/r^3(t) \end{cases}$，其中，$r(t) = \sqrt{u^2(t) + v^2(t)}$，且 $u(0) = 1, \dot{u}(0) = 2, \dot{v}(0) = 2, v(0) = 1$，试选择一组状态变量，将其变换成 MATLAB 语言能直接求解的微分方程组形式，并绘制出 $u(t), v(t)$ 的轨迹曲线。

14 已知微分方程可以表示为[17] $\begin{cases} \dot{u}_1 = u_3 \\ \dot{u}_2 = u_4 \\ 2\dot{u}_3 + \cos(u_1 - u_2)\dot{u}_4 = -\mathrm{g}\sin u_1 - \sin(u_1 - u_2)u_4^2 \\ \cos(u_1 - u_2)\dot{u}_3 + \dot{u}_4 = -\mathrm{g}\sin u_2 + \sin(u_1 - u_2)u_3^2 \end{cases}$
其中，$u_1(0) = 45, u_2(0) = 30, u_3(0) = u_4(0) = 0$，g= 9.81，试求解此微分方程，并绘制出各个状态变量的时间曲线。

15 试求出隐式微分方程 $\begin{cases} \dot{x}_1\ddot{x}_2\sin(x_1x_2) + 5\ddot{x}_1\dot{x}_2\cos(x_1^2) + t^2x_1x_2^2 = \mathrm{e}^{-x_2^2} \\ \ddot{x}_1x_2 + \ddot{x}_2\dot{x}_1\sin(x_1^2) + \cos(\ddot{x}_2x_2) = \sin t \end{cases}$ 的数值解，$x_1(0) = 1, \dot{x}_1(0) = 1, x_2(0) = 2, \dot{x}_2(0) = 2$，并绘制出轨迹曲线。

16 下面的方程在传统微分方程教程中经常被认为是刚性微分方程。试用常规微分方程解法和刚性微分方程解法分别求解这两个微分方程的数值解，并求出解析解，用状态变量曲线比较数值求解的精度。

① $\begin{cases} \dot{y}_1 = 9y_1 + 24y_2 + 5\cos t - \dfrac{1}{3}\sin t,\ y_1(0) = \dfrac{1}{3} \\ \dot{y}_2 = -24y_1 - 51y_2 - 9\cos t + \dfrac{1}{3}\sin t,\ y_2(0) = \dfrac{2}{3} \end{cases}$

② $\begin{cases} \dot{y}_1 = -0.1y_1 - 49.9y_2,\ y_1(0) = 1 \\ \dot{y}_2 = -50y_2,\ y_2(0) = 2 \\ \dot{y}_3 = 70y_2 - 120y_3,\ y_3(0) = 1 \end{cases}$

17 考虑下面的化学反应系统的反应速度方程组 $\begin{cases} \dot{y}_1 = -0.04y_1 + 10^4y_2y_3 \\ \dot{y}_2 = 0.04y_1 - 10^4y_2y_3 - 3\times10^7y_2^2 \\ \dot{y}_3 = 3\times10^7y_2^2 \end{cases}$，
其初值为 $y_1(0) = 1, y_2(0) = y_3(0) = 0$，该方程往往被认为是刚性方程。试采用 `ode45()` 对之求解，观察是否能正确求解，如果不能求解应该如何解决问题？

18 试求出习题 4 中给出的微分方程边值问题数值解，绘制出 $y(t)$ 曲线，并和该习题得出的解析解比较精度。

19 考虑 Van der Pol 方程 $\ddot{y} + \mu(y^2 - 1)\dot{y} + y = 0$，试求解 $\mu = 1$，且边值 $y(0) = 1, y(5) = 3$ 时方程的数值解。如果假设 μ 为自由参数，试求出满足边值条件，且满足 $\dot{y}(5) = -2$ 时方程的数值解及 μ 的值，并绘图验证之。

20 考虑简单的线性微分方程 $y^{(4)} + 3y^{(3)} + 3\ddot{y} + 4\dot{y} + 5y = \mathrm{e}^{-3t} + \mathrm{e}^{-5t}\sin(4t + \pi/3)$，且方程的初值为 $y(0) = 1, \dot{y}(0) = \ddot{y}(0) = 1/2, y^{(3)}(0) = 0.2$，试用 Simulink 搭建起

系统的仿真模型，并绘制出仿真结果曲线。考虑上面的模型，假设给定的微分方程变化成时变线性微分方程 $y^{(4)}+3ty^{(3)}+3t^2\ddot{y}+4\dot{y}+5y=\mathrm{e}^{-3t}+\mathrm{e}^{-5t}\sin(4t+\pi/3)$，而方程的初值仍为 $y(0)=1,\ \dot{y}(0)=\ddot{y}(0)=1/2, y^{(3)}(0)=0.2$，试用 Simulink 搭建起系统的仿真模型，并绘制出仿真结果曲线。

21 考虑延迟微分方程 $y^{(4)}(t)+4y^{(3)}(t-0.2)+6\ddot{y}(t-0.1)+6\ddot{y}(t)+4\dot{y}(t-0.2)+y(t-0.5)=\mathrm{e}^{-t^2}$，且在 $t\leqslant 0$ 时该方程具有零初始条件，试分别用 Simulink 建模与 dde23() 函数求解的方式直接求解该微分方程，并绘制出 $y(t)$ 曲线。

22 假设已知误差信号 $e(t)$，试构造出求取 ITAE, ISE, ISTE 准则的封装模块。要求：误差信号 $e(t)$ 为该模块的输入信号，双击该模块弹出一个对话框，允许用户用列表框的方式选择输出信号形式，将选定的 ITAE, ISE, ISTE 之一作为模块的输出端显示出来。其中 ISE 误差准则定义为 $J_{\mathrm{ISE}}=\displaystyle\int_0^{\infty}e^2(t)\mathrm{d}t$，ITAE 准则定义为 $J_{\mathrm{ITAE}}=\displaystyle\int_0^{\infty}t|e(t)|\mathrm{d}t$，而 ISTE 的定义为 $J_{\mathrm{ISTE}}=\displaystyle\int_0^{\infty}t^2e^2(t)\mathrm{d}t$。

23 建立起如图 4-49 所示非线性系统[18]的 Simulink 框图，并观察在单位阶跃信号输入下系统的输出曲线和误差曲线。

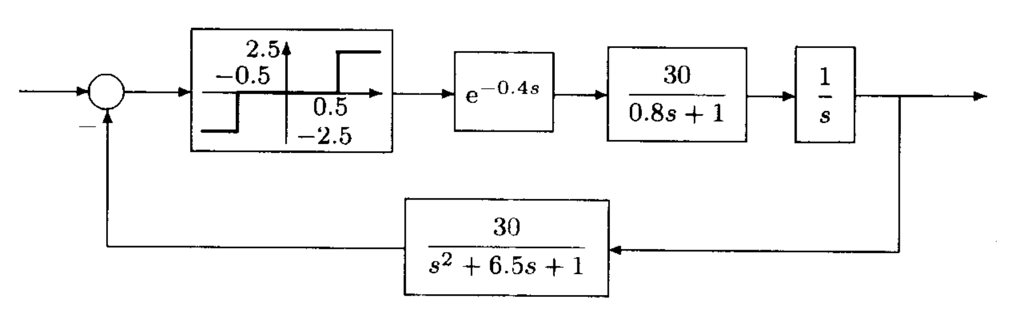


图 4-49 习题 23 的系统方框图

24 建立起如图 4-50 所示非线性系统[19]的 Simulink 框图，并设阶跃信号的幅值为 1.1，观察在阶跃信号输入下系统的输出曲线和误差曲线。求取系统在阶跃输入下的工作点，并在工作点处对整个系统矩形线性化，得出近似的线性模型。对近似模型仿真分析，将结果和精确仿真结果进行对比分析。另外，本系统中涉及到两个非线性环节的串联，试问这两个非线性环节可以互换吗？试从仿真结果上加以解释。

25 试构造出如图 4-51 所示的 Simulink 仿真框图的数学公式。

26 考虑下面给出的延迟微分方程模型 $\mathrm{d}y(t)/\mathrm{d}t=\dfrac{0.2y(t-30)}{1+y^{10}(t-30)}-0.1y(t)$，假设 $y(0)=0.1$，试用 Simulink 搭建仿真模型，并对该系统进行仿真，绘制出 $y(t)$ 曲线。

27 韩京清研究员在自抗扰控制器中首先提出了扩张的状态观测器 (extended state

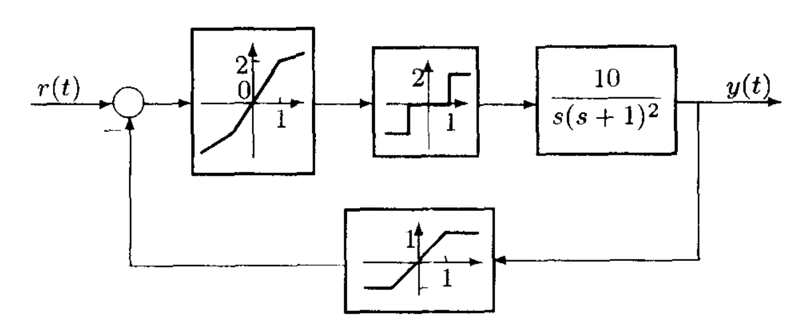


图 4-50 习题 24 的非线性系统方框图

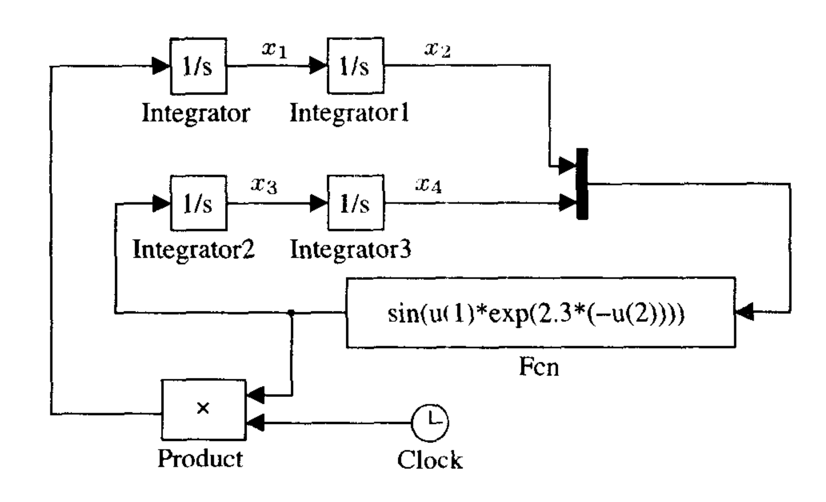


图 4-51 习题 25 的 Simulink 仿真框图

observer，简称为 ESO）[15]，其数学表示为

$$\begin{cases} z_1(k+1) = z_1(k) + T[z_2(k) - \beta_{01}e(k)] \\ z_2(k+1) = z_2(k) + T[z_3(k) - \beta_{02}\text{fal}(e(k), 1/2, \delta) + bu(k)] \\ z_3(k+1) = z_3(k) - T\beta_{03}\text{fal}(e(k), 1/4, \delta) \end{cases} \tag{4-6-1}$$

其中 $e(k) = z_1(k) - y(k)$，且 $\text{fal}(e, a, \delta) = \begin{cases} e\delta^{a-1}, & |e| \leqslant \delta \\ |e|^a \text{sgn}(e), & |e| > \delta \end{cases}$，和普通状态观测器一样，该观测器接受 $u(k)$ 和 $y(k)$ 为其输入信号。根据该数学模型，这里，$y(k)$ 实际上是第 2 路模块输入信号。试编写出其对应的 S-函数。

28 由给定的微分–跟踪器和扩张状态观测器，可以根据下面的数学工具构造起自抗扰控制器

$$\begin{cases} e_1 = v_1(k) - z_1(k), \quad e_2 = v_2(k) - z_2(k) \\ u_0 = \beta_1\text{fal}(e_1, a_1, \delta_1) + \beta_2\text{fal}(e_2, a_2, \delta_1) \\ u(k) = u_0 - z_3(k)/b \end{cases} \tag{4-6-2}$$

可以看出，该控制器算法中不存在动态的过程，故可以设置连线和离散的状态个数均为 0。试根据上面的公式编写出相应的 S-函数。

参考文献

[1] Forsythe G E, Malcolm M A, Moler C B. Computer methods for mathematical computations [M]. Englewood Cliffs: Prentice-Hall, 1977

[2] Bogdanov A. Optimal control of a double inverted pendulum on a cart [R]. Technical Report CSE-04-006, Department of Computer Science & Electrical Engineering, OGI School of Science & Engineering, OHSU, 2004

[3] 武汉大学，山东大学. 计算方法 [M]. 北京: 人民教育出版社, 1979

[4] 张庆灵、杨冬梅. 不确定广义系统的分析与综合 [M]. 沈阳: 东北大学出版社, 2003

[5] Liberzon D, Morse A S. Basic problems in stability and design of switched systems [J]. IEEE Control Systems Magazine, 1999, 19(5):59~70

[6] 孙增圻，袁曾任. 控制系统的计算机辅助设计 [M]. 北京: 清华大学出版社, 1988

[7] Åström K J. Introduction to stochastic control theory [M]. London: Academic Press, 1970

[8] Shampine L F, Kierzenka J, Reichelt M W. Solving boundary value problems for ordinary differential equation problems in MATLAB with bvp4c [Z], 2000

[9] 顾启泰. 系统设计与仿真 [M]. 北京: 清华大学出版社, 1995

[10] The MathWorks Inc. Simulink˜user's manual [Z], 2005

[11] 薛定宇，陈阳泉. 基于 MATLAB/Simulink 的系统仿真技术与应用 [M]. 北京: 清华大学出版社, 2002

[12] 薛定宇. 控制系统计算机辅助设计 —— MATLAB 语言与应用 (第 2 版) [M]. 北京: 清华大学出版社, 2006

[13] Xue D, Atherton D P. Simulation analysis of continuous systems driven by Gaussian white noise. Jamshidi M, Herget C J, eds., Recent Advances in Computer Aided Control Systems Engineering. Amsterdam: Elsevier Science Publishers B V, 1992

[14] 薛定宇. 一类非线性系统 Fokker-Planck 系统的解法 [J]. 自动化学报, 1996, 22(3):323~331

[15] 韩京清，袁露林. 跟踪微分器的离散形式 [J]. 系统科学与数学, 1999, 19(3):268~273

[16] 张化光，王智良，黄伟. 混沌系统的控制理论 [M]. 沈阳: 东北大学出版社, 2003

[17] Molor C B. Numerical Computing with MATLAB [M]. MathWorks Inc, 2004

[18] 刘德贵，费景高. 动力学系统数字仿真算法 [M]. 北京: 科学出版社, 2001

[19] 王万良. 自动控制原理 [M]. 北京: 科学出版社, 2001

第 5 章 最优化问题的计算机求解

方程求解问题是科学与工程研究中经常遇到的问题。线性代数方程可以用第 4 章中介绍的方法直接求解，非线性方程或一般多项式方程的求解将在 5.1 节中介绍，在该节中将先介绍一元或二元方程的图解法，然后介绍基于 MATLAB 符号运算工具箱的多项式方程或一类可以化成多项式方程的代数方程的解析解方法，再介绍一般非线性方程组的数值解法。该节还将介绍方程全局问题在控制研究中的应用，如控制系统的根轨迹分析方法、Lambert 函数在系统稳定边界中的应用，并研究各种 Riccati 变形方程的数值方法等。

最优化技术是当前科学研究中的一类重要的手段。所谓最优化就是找出使得目标函数值达到最小或最大的自变量值的方法。从其分类看有无约束最优化问题和有约束最优化问题。5.2 节介绍无约束最优化问题以及 MATLAB 求解方法，图解法和一般的数值算法，引入全局最优解与局部最优解的概念，并通过例子给出基于 ITAE 指标的最优控制概念，提出基于 MATLAB 最优化技术的求解方法。5.3 节介绍有约束最优化的概念，引入约束可行区域的概念，并就线性规划问题、二次型规划问题和一般非线性规划问题的 MATLAB 语言求解进行详细的介绍，还将介绍有约束最优化问题在最优控制器设计中的应用。5.4 节进一步引申最优化问题，引入整数规划的概念，介绍如何用 MATLAB 语言求解混合整数线性规划问题，并介绍基于“分枝定界法”的一般整数规划问题及其 MATLAB 实现。5.5 节还将通过两个最优控制问题求解程序——OCD 程序和 Riots 程序，介绍最优控制问题的求解方法，并探讨 Minimax 问题在不确定性系统最优控制器设计问题。

通过本章内容的学习，读者应该能掌握一般非线性方程及非线性最优化问题的实际求解方法。

5.1 代数方程的求解

方程求解是控制系统研究中经常遇到的问题。本节先介绍简单代数方程的图解方法，然后介绍多项式类方程的准解析解方法，还将介绍一般非线性方程的数值解方法。本节还将研究方程求解在控制中的具体应用，包括线性系统的根轨迹研究，延迟系统的稳定区域研究，并较深入地研究各种扩展的 Riccati 方程数值解方法，还将探讨基于方程求解的隐式微分方程求解方法。

5.1.1 代数方程的图解法

MATLAB 提供了很强的一元、二元隐函数绘制功能，充分利用这些功能就可以将一元、二元的方程用曲线表示，并由曲线的交点读出方程的实数根来。然而，方程的图解法是有局限性的，仅适用于一元、二元方程，多元方程是不能用图解法直接求解的。本小节将通过例子演示一元、二元方程的求根问题。

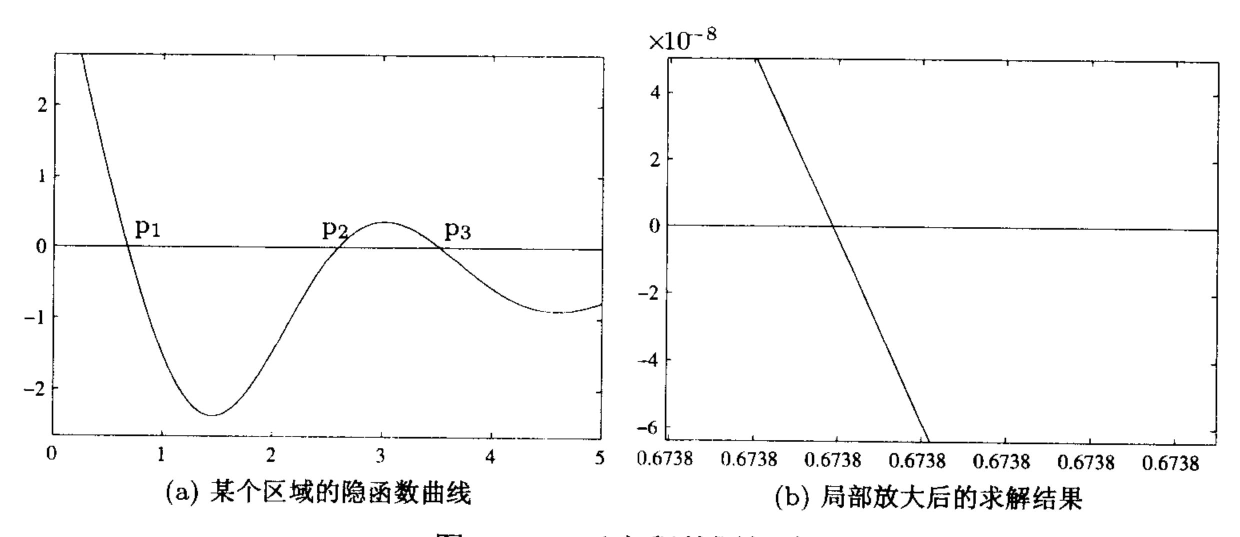
1. 一元方程的图解法

用 `ezplot()` 函数可以绘制出给定的隐函数 $f(x)=0$ 曲线，所以可以用图解法从给出的曲线和 $y=0$ 线的交点上读出所有的实数解。

例 5-1 用图解法求解方程 $\mathrm{e}^{-3t}\sin(4t+2)+4\mathrm{e}^{-0.5t}\cos(2t)=0.5$。

求解 用 `ezplot()` 函数可以绘制出如图 5-1 (a) 所示的曲线，该曲线与横轴的所有交点均是原一元方程的解。

```
>> ezplot('exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5',[0 5])
   hold on, line([0,5],[0,0])% 同时绘制横轴
```



(a) 某个区域的隐函数曲线　　(b) 局部放大后的求解结果

图 5-1 一元方程的图解法

从得出曲线可以看出，该方程可能有多个实根，如果想用图解法得出某个根，则可以对该点附近局部放大，直到曲线穿越 0 线，且 t 轴给出的各个标点的数值完全一

致时，则可以断定原方程的解即为 t 轴标度，如图 5-1 (b) 所示，即该方程的一个解为 $t=0.6738$。将其代入方程则

```
>> syms x; t=0.6738;
   vpa(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
```

得出解的误差为 -0.2985×10^{-3}。故可见得出的根是原方程的根，但精度不是很高。用类似的方法还可以得出并验证其他的解。

2. 二元方程的图解法

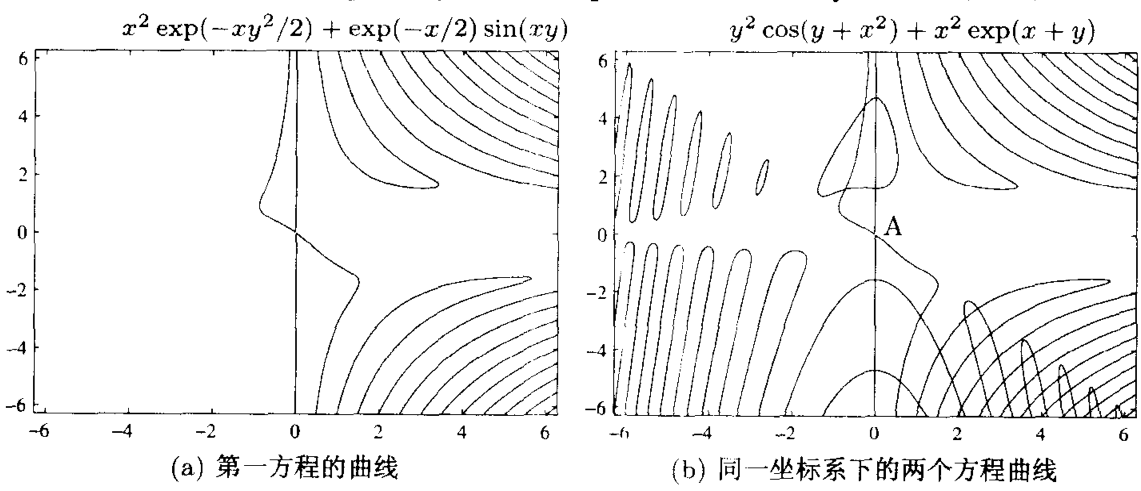
二元方程也是可以通过图解法求解的，可以通过 ezplot() 函数将第一个方程对应的曲线绘制出来，再使用 hold on 命令保持图形不被刷新，最后调用 ezplot() 函数将第二条曲线在同一坐标系下绘制出来。得出曲线后就可以通过读取交点坐标的方式得出联立方程的根。

例 5-2 用图解法求解下面的联立方程。

$$\begin{cases} x^2\mathrm{e}^{-xy^2/2}+\mathrm{e}^{-x/2}\sin(xy)=0\\ x^2\cos(x+y^2)+y^2\mathrm{e}^{x+y}=0\end{cases}$$

求解 利用隐函数图形绘制的方法，可以用图解法直接求解二元方程组，则可以通过下面的语句绘制出第一个方程的曲线，如图 5-2 (a) 所示。

```
>> ezplot('x^2*exp(-x*y^2/2)+exp(-x/2)*sin(x*y)') % 第一个方程曲线
```



(a) 第一方程的曲线　　(b) 同一坐标系下的两个方程曲线

图 5-2 联立方程图解法示意图

该曲线上所有的点均满足第一个方程。可以用 hold on 语句保护当前的坐标系，再用 ezplot() 函数绘制第二个方程的曲线，这样在同一坐标系下绘制出两条方程曲线，如图 5-2 (b) 所示。

```
>> hold on % 保护当前坐标系
   ezplot('y^2 *cos(y+x^2) +x^2*exp(x+y)')
```

这两个方程对应曲线的交点就是联立方程的解，所以可以通过图解法来求取二元

联立方程的全部实根。但应该指出，这样求出的交点可能有的不是方程的根。观察图 5-2 (a) 可见，满足第一方程的曲线自身就有交点，该交点不一定能满足第二方程，所以图 5-2 (b) 中的 A 点不是方程的根。

5.1.2 多项式型方程的准解析解法

在介绍多项式方程的一般解法之前，先考虑下面给出的一个例子。

例 5-3 试用图解方法求解下面的二元方程 $\begin{cases} x^2+y^2-1=0 \\ 0.75x^3-y+0.9=0 \end{cases}$。

求解 用图解方法，可以用下面的语句直接绘制出两条曲线，如图 5-3 所示。曲线的交点就是原联立方程的解。

```
>> ezplot('x^2+y^2-1'); hold on  % 绘制第一方程的曲线
   ezplot('0.75*x^3-y+0.9')      % 绘制第二方程的曲线
```

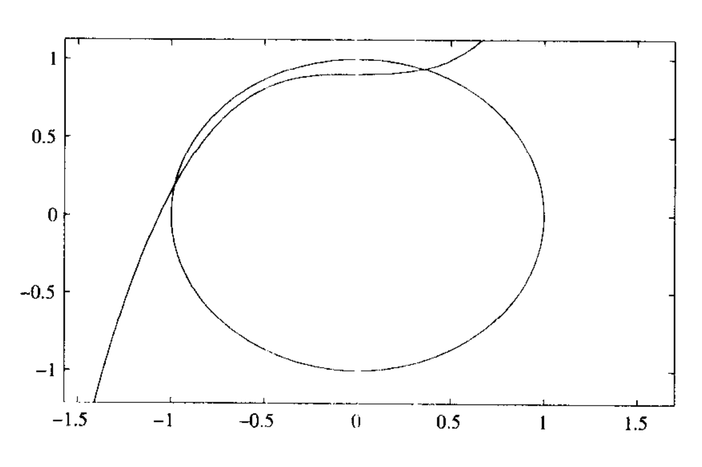


图 5-3 联立方程图解法示意图

从图 5-3 可见，这两条曲线共有两个交点，故可能轻易得出结论，原联立方程有两对根。事实上，这样的结论是错误的，由第 2 个方程可以显式地将 y 写成 x^3 的形式，代入第 1 方程则得出关于 x 的 6 次多项式方程，该方程应该有 6 对根。因为用二维图形只能求解出方程的实根，而不能求解出方程的复数根，所以利用图解法有时也可能得出错误的结论。

一般多项式方程的根可以为实数，也可以为复数。MATLAB 符号运算工具箱中给出的 solve() 函数对多项式类方程是十分有效的，可以用该函数求解出多项式方程所有的根。该函数的定义格式为

```
S=solve(eqn1,eqn2,⋯,eqnn)                    % 最简调用方式
[x,⋯]=solve(eqn1,eqn2,⋯,eqnn)                % 直接得出根
[x,⋯]=solve(eqn1,eqn2,⋯,eqnn,'x,⋯')          % 同上，并指定变量
```

其中，eqn_i 为第 i 个方程的符号表达式，可以同时求解若干个联立方程，还可以

按照第 3 种调用格式显式地指出需要求解方程的变量名，第 2、3 种调用格式将求出方程的根，并按各个变量名返回到 MATLAB 的工作空间，第 1 种调用格式将返回一个结构体型变量 S，其各个成员变量，如 S.x 表示方程的根。

例 5-4 试用 solve() 函数求取例 5-3 中给出的联立方程。

求解 考虑上述的方法，利用 solve() 函数在 MATLAB 中可以给出如下的命令:

```
>> syms x y;
   [x,y]=solve('x^2+y^2-1=0','75*x^3/100-y+9/10=0')
```

这样可以得出原方程的 6 对根为

$$\boldsymbol{x}=\begin{bmatrix} .35696997189122287798839037801365 \\ .86631809883611811016789809418 65\pm 1.2153712664671427801318378544 39\mathrm{j} \\ -.55395176056834560077984413882 74\pm.35471976465080793456863789934 94\mathrm{j} \\ -.98170264842676789676449828873194 \end{bmatrix}$$

$$\boldsymbol{y}=\begin{bmatrix} .93411585960628007548796029415446 \\ -1.4916064075658223174787216959 26\pm.70588200721402267753918827138 84\mathrm{j} \\ .92933830226674362852985276677 20\pm.21143822185895923615623381762 21\mathrm{j} \\ .19042035099187730240977756415289 \end{bmatrix}$$

显然，这里得出的方程阶次太高，不存在解析解。然而，利用 MATLAB 的符号运算工具箱可以得出原始问题的高精度数值解，故这里称之为准解析解。可以看出，除了前面得出的两组实数根外，还得出了另外 4 组复数根，这是用普通数值解法所得不出来的。下面验证一下这样得出的根是不是原方程的根。直接给出下面的语句则可以发现，误差矩阵的范数为 7.2118×10^{-31}，所以得出的根基本满足原方程。

```
>> norm(double([eval('x.^2+y.^2-1') eval('75*x.^3/100-y+9/10')]))
```

例 5-5 多元多项式方程也可以用 solve() 函数直接求解。试求解下面的联立方程

$$\begin{cases} x+3y^3+2z^2=1/2 \\ x^2+3y+z^3=2 \\ x^3+2z+2y^2=2/4 \end{cases}$$

求解 给出的联立方程是关于 x,y,z 的三元联立方程，从方程本身可见它只含有多项式项，所以从理论上说可以将之转换成一元的多项式方程，故方程可以由下面的 MATLAB 语句求出高精度数值解。

```
>> [x,y,z]=solve('x+3*y^3+2*z^2=1/2','x^2+3*y+z^3=2',...
                 'x^3+2*z+2*y^2=2/4')
```

事实上，该方程最终被变换成 27 次的多项式方程，所以得出的解向量 $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ 均

为有 27 个分量的向量。由于篇幅所限，在这里不列出全部的解，只列出其中一个根。

$$\begin{cases} x_1 = -1.0869654762986136074917644096117 \\ y_1 = 0.037264668450644375527750811297 21 \\ z_1 = 0.89073290972562790151300874796949 \end{cases}$$

可以用下面的语句验证解的误差为 6.9146×10^{-26}，故而得出的结果是很精确的。

```
>> err=[x+3*y.^3+2*z.^2-1/2,x.^2+3*y+z.^3-2,x.^3+2*z+2*y.^2-2/4];
   norm(double(eval(err)))
```

其实，方程中若干式子也可以写成多项式乘积的形式。例如，联立方程最后一个式子改写成 $x^3+2zy^2=2/4$，其中含有非线性项 zy^2，这样的方程也能通过 solve() 函数直接求解，这时只需将求解语句变为

```
>> [x,y,z]=solve('x+3*y^3+2*z^2=1/2',...
                 'x^2+3*y+z^3=2','x^3+2*z*y^2=2/4');
   err=[x+3*y.^3+2*z.^2-1/2,x.^2+3*y+z.^3-2,x.^3+2*z.*y.^2-2/4];
   norm(double(eval(err)))    % 将解代入求误差
```

精度为 4.3077×10^{-26}。

例 5-6 试求解下面的方程，其中含有自变量的倒数等形式。

$$\begin{cases} \dfrac{1}{2}x^2+x+\dfrac{3}{2}+2\dfrac{1}{y}+\dfrac{5}{2y^2}+3\dfrac{1}{x^3}=0 \\ \dfrac{y}{2}+\dfrac{3}{2x}+\dfrac{1}{x^4}+5y^4=0 \end{cases}$$

求解 这样的方程指望求取解析解基本上是不可能的，但用下面的语句可以直接得出原方程的精确数值解，共有 26 对根。

```
>> syms x y;
   [x,y]=solve('x^2/2+x+3/2+2/y+5/(2*y^2)+3/x^3=0',...
           'y/2+3/(2*x)+1/x^4+5*y^4','x,y'); size(x)
```

其中，第 1 对根为

$$\begin{cases} x_1 = 0.93020851860976459141084889837 \pm \mathrm{j}0.44242013491916995256639798767 \\ y_1 = -0.39388334385234983573088749775 \pm \mathrm{j}0.62771855170677843775952969855 \end{cases}$$

将得出的全部方程根代入原始方程，则能得出很小的计算误差，达到 10^{-30} 数量级，说明该方程各个解的正确性。

```
>> err=['x.^2/2+x+3/2+2./y+5./(2*y.^2)+3./x.^3,y/2+',...
        '3./(2*x)+1./x.^4+5*y.^4'];
   norm(double(eval(err)))
```

例 5-7 试求解下面带有参数的方程 $\begin{cases} x^2+ax^2+6b+3y^2=0 \\ y=a+x+3 \end{cases}$。

求解 MATLAB 符号运算工具箱中提供的 solve() 函数还可以直接实现带有变量的方程的解，这样的求解用普通的数值解方法是不能实现的。求解上述方程只需给出下面语句即可。

```
>> syms a b x y;
   [x,y]=solve('x^2+a*x^2+6*b+3*y^2=0','y=a+(x+3)','x,y')
```

可以得出方程组的解为

$$x=\frac{-6a-18\pm2\sqrt{-21a^2-45a-27-24b-6ab-3a^3}}{2(4+a)},\quad y=a+x+3$$

其实，该方法同样适用于更高阶方程的解，但得出的解很冗长，不适合显示出来。

然而，解析求解的方法并不是万能的，因为这里的例子最终可以转换为一元多项式方程，所以能用它求解，但更一般的非线性方程是不能解出的，只能求出其一个数值解。

5.1.3 一般非线性方程数值解

MATLAB 语言环境中提供了 fsolve() 函数，能够求出已知多元方程的一个实数根。该函数的调用格式为

```
x=fsolve(Fun,x0)                                  % 最简求解语句
[x,f,flag,out]=fsolve(Fun,x0,opt,p1,p2,···)      % 一般求解格式
```

其中，Fun 为描述需求解的方程 M-函数、inline() 函数或匿名函数，x_0 为搜索点的初值，方程求根程序将从该值开始以逐步减小误差的算法搜索出满足方程的实根 x。若返回的 flag 大于 0，则表示求解成功，否则求解出现问题。

对于更复杂的问题，用户可以定义方程求解控制参数 opt 来控制求解方法或其他要求，更好地得出方程的根。该变量是一个结构体数据，其常用的成员变量在表 5-1 中给出，用户可以用下面的语句修改控制变量。

```
opt=optimset; %获得默认的常用变量
opt.TolX=1e-10; 或 set(opt,'TolX',1e-10) % 修改参数
```

其中的某些值，如 MaxFunEvals 和问题类型有关，如方程求解和有约束最优化问题一般选择其值为 100 倍的自变量个数，而无约束最优化问题一般支持 200 倍的变量个数。

例 5-8 Lambert 函数是一类特殊函数，该方程的数学表示为 $u=\mathscr{U}(x)$，其中，x 为自变量，u 为 Lambert 方程 $ue^u=x$ 的解，对不同的 x 值，试求解 Lambert 方程得出

表 5-1 方程求解与最优化的控制参数表

参数名	参 数 说 明
Display	中间结果显示方式，其值可以取 off 表示不显示中间值，iter 表示逐步显示，notify 表示在求解不收敛时给出提示，final 只显示最终值
GradObj	求解最优化问题时使用，表示目标函数的梯度是否已知，可以选择为 'off' 或 'on'
LargeScale	表示是否使用大规模问题算法，取值为 on 或 off，变量较少的问题不必采用该算法
MaxIter	方程求解和优化过程最大允许的迭代次数，若方程未求出解，可以适当增加该值
MaxFunEvals	方程函数或目标函数的最大调用次数
TolFun	误差函数误差限控制量，当函数的绝对值小于此值即终止求解
TolX	解的误差限控制量，当解的绝对值小于此值即终止求解

u 值，并绘制出 x 和 u 之间的关系曲线。

求解 通过循环的方法对不同的 x 值进行求解，就可以绘制出 Lambert 函数的曲线，如图 5-4 所示。

```
>> xx=0:.05:10; x0=0; h=optimset; h.Display='off'; y=[];
   for x=xx,
      f=@(u)[u.*exp(u)-x]; y1=fsolve(f,x0,h); x0=y1; y=[y,y1];
   end
   plot(xx,y);
```

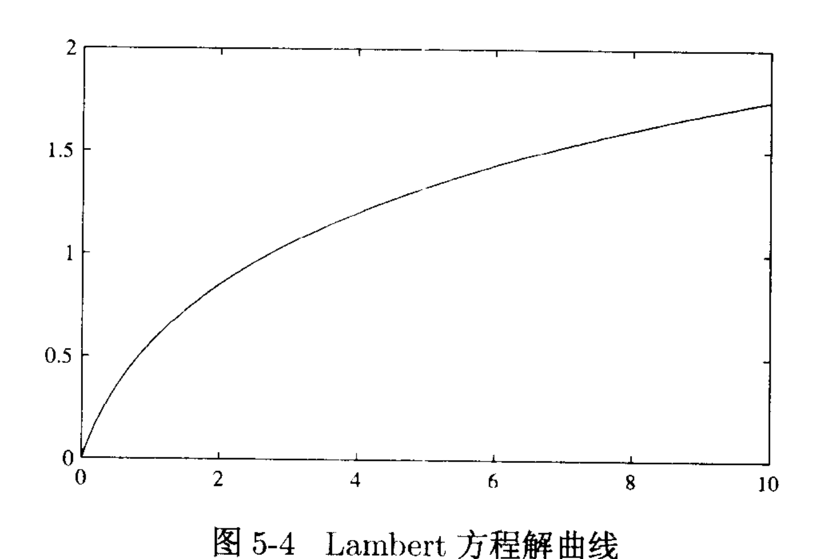


图 5-4 Lambert 方程解曲线

MATLAB 的符号运算工具箱中提供了一个 Lambert 函数求值函数 `lambertw()`，调用格式为 `u=lambertw(x)`。可以用来直接求取 Lambert 函数的值。用下面语句可以直接绘制出与图 5-4 完全一致的曲线。

```
>> y0=lambertw(xx); plot(xx,y0)
```

例 5-9 试用数值方法求解例 5-3 中给出的二元方程。

求解 令 $x_1 = x, x_2 = y$，可以用匿名函数描述此二元方程。选择初值 $x_0 = 1, y_0 = 2$，则调用 fsolve() 函数可以直接求解原方程

```
>> f=@(x)[x(1)^2+x(2)^2-1; 0.75*x(1)^3-x(2)+0.9];
   [x,Y,c,d]=fsolve(f,[1; 2]),
```

则可以直接求出方程的根为 $\boldsymbol{x} = [0.35697, 0.93412]^{\mathrm{T}}$，两个方程的误差达到 10^{-9} 数量级。由返回的 d 变量可知迭代次数为 6，模型函数调用次数为 21。

若改变初始猜测值，令 $\boldsymbol{x}_0 = [-1, 0]^{\mathrm{T}}$，则

```
>> [x,Y,c,d]=fsolve(f,[-1,0]',OPT); x, Y, kk=d.funcCount
```

这样可以搜索出方程的解为 $\boldsymbol{x} = [-0.98170, 0.1904]^{\mathrm{T}}$，误差仍然是 10^{-10} 数量级。

可见，初值改变之后，还能得出另外一组解。所以初值的选择有时对整个问题的求解有很大的影响，在某些初值下甚至无法搜索到方程的解。

例 5-10 重新考虑例 5-1 中给出的方程 $\mathrm{e}^{-3t}\sin(4t+2) + 4\mathrm{e}^{-0.5t}\cos(2t) = 0.5$，试用数值方法求其更精确的数值解。

求解 由于原方程是非线性的，所以没有解析解。例 5-1 中用图解法得出了一个数值解为 $t = 3.5203$，以此解为初值，则可以用 fsolve() 函数得出更精确的数值解。

先使用符号运算工具箱中的 solve() 函数求解此问题。

```
>> syms t x;
   solve(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
```

利用返回因式可以得到精确解 $x = 0.67374570500134756702960220427474$。可见，这样的方法不允许手工选择初始搜索点，所以只能获得这样的结果。用图解法自由地选择初值，再用数值解法可以得出任意初值下的数值解。

```
>> y=@(t)exp(-3*t).*sin(4*t+2)+4*exp(-0.5*t).*cos(2*t)-0.5;
   ff=optimset; ff.Display='iter'; [t,f]=fsolve(y,3.5203,ff)
```

可以得出 $t = 3.52026389294877, y = -6.8\times10^{-10}$。代入原方程发现，该解的精度明显高于图解法的精度，且可以任意选择初值。从算法本身看，默认的求解精度偏低，所以可以试着重新设置相关的控制变量，就可以得出下面更精确的结果。

```
>> ff=optimset; ff.TolX=1e-16; ff.TolFun=1e-30;
   ff.Display='iter'; [t,f]=fsolve(y,3.5203,ff)
```

经过 2 次迭代，则可以得出 $t = 3.52026389244155, y = 0$。从求解效果看，给出一个很精确的初值对整个求解速度和精度没有太大的帮助，真正使得本例子获得高精度解的还是 TolFun 属性。

5.1.4 方程求解在控制系统研究中的应用

和其他工程系统相近，控制理论研究中也大量使用代数方程求解技术。例如，在控制理论中重要的根轨迹就是建立在方程求解概念上的，但由于早期缺乏数值求解的工具，经典控制理论中只能考虑根轨迹示意图的绘制方式。有了强大的计算机工具，绘制精确的根轨迹轻而易举，也就没有必要采用示意图的方法绘制根轨迹了。另外，现代代数 Riccati 方程求解算法是建立在一种特殊解法前提下的，这里将介绍基于搜索方法的 Riccati 代数方程求解方法，还将得出以往不能得出的结果。

1. 线性系统稳定区域研究 —— 根轨迹方法

若已知无延迟系统模型 $G(s)=N(s)/D(s)$，则在比例控制 K_p 下，单位负反馈闭环系统的稳定性可以由特征方程 $1+K_\mathrm{p}G(s)=0$ 时根的位置判定，亦即由多项式方程 $D(s)+K_\mathrm{p}N(s)=0$ 的根来判定。令 $K_\mathrm{p}=0$，则可以求出系统的特征根初始值，逐渐增大 K_p 的值，系统的特征根位置也发生变化，这样可以定义每个特征根随 K_p 变化的轨迹，称为系统的根轨迹曲线。

对每个 K_p 求解多项式方程则可以得出相应的特征根位置，根据不同 K_p 值解出的特征根位置，则可以追踪出系统的根轨迹。MATLAB 控制系统工具箱中的 `rlocus()` 函数可以直接绘制系统的根轨迹，该函数的调用格式为

```
rlocus(G)    % 不返回变量将自动绘制根轨迹曲线
rlocus(G,K)    % 给定增益向量，绘制根轨迹曲线
[R,K]=rlocus(G)    % R 为闭环特征根构成的复数矩阵
rlocus(G1,'-',G2,'-.b',G3,':r')    % 同时绘制若干系统的根轨迹
```

其中 G 为单变量系统的数学模型，可以为离散模型。

例 5-11 试绘制出开环系统 $G(s)=\dfrac{1}{s^4+8s^3+36s^2+80s}$ 的根轨迹曲线，并指出使得单位负反馈下闭环系统稳定的增益范围。

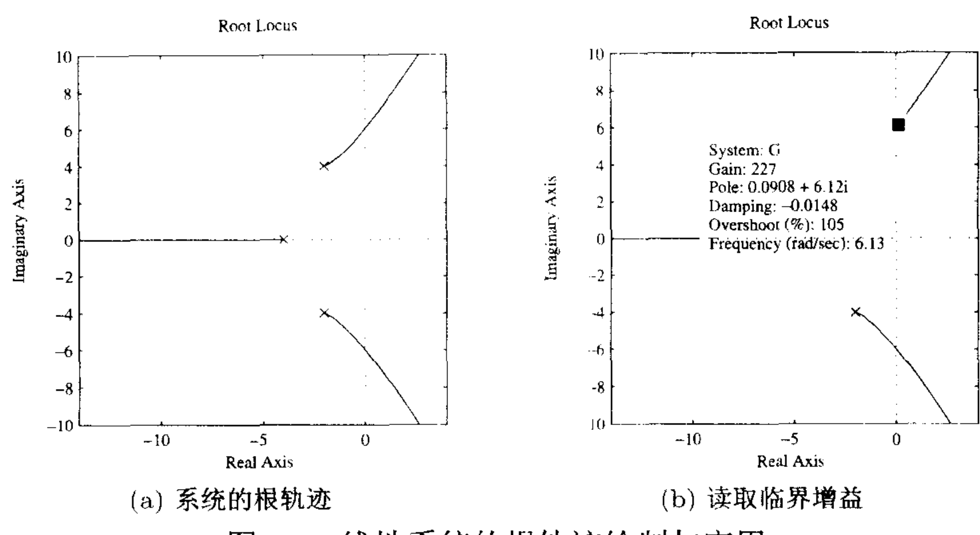
求解 如果不采用计算机工具，直接采用控制理论中介绍的示意图方法则无法绘制此系统的根轨迹，因为高阶系统的零极点是未知的，无法确定根轨迹的起点和终止点。利用 MATLAB 控制系统工具箱中的 `rlocus()` 函数，则可以容易地绘制出系统的根轨迹，如图 5-5 (a) 所示。

```
>> G=tf(1,[1 8 36 80]); rlocus(G)
```

求取临界增益通常是控制理论课程中较难的任务，但绘制了根轨迹曲线，在 MATLAB 环境下求解增益的问题是轻而易举的，因为单击临界点，则可以得出如图 5-5 (b) 所示的显示结果，即临界增益为 227。

2. Lambert 函数在延迟微分方程稳定域研究中的应用

若系统含有延迟，则系统的特征方程 $1+K_\mathrm{p}(s)\mathrm{e}^{-Ts}=0$ 不再为多项式方程，



(a) 系统的根轨迹　　(b) 读取临界增益

图 5-5　线性系统的根轨迹绘制与应用

其求解变得很困难，因为当前不存在任何算法确保求出方程的所有特征根。这样只能求解某些特殊类型系统的稳定区域问题[1]。

先考虑一个简单的延迟微分方程 $\dot{y}(t)=ay(t-1)$，方程两边进行 Laplace 变换，则可以直接得出 $sY(s)=a\mathrm{e}^{-s}Y(s)$，显然可以得出 $s\mathrm{e}^s=a$，该式满足 Lambert 方程，其解可以表示为 $s=\mathscr{W}(a)$。可以看出，若想使系统稳定，则 $s<0$，即 $\mathscr{W}(a)<0$。

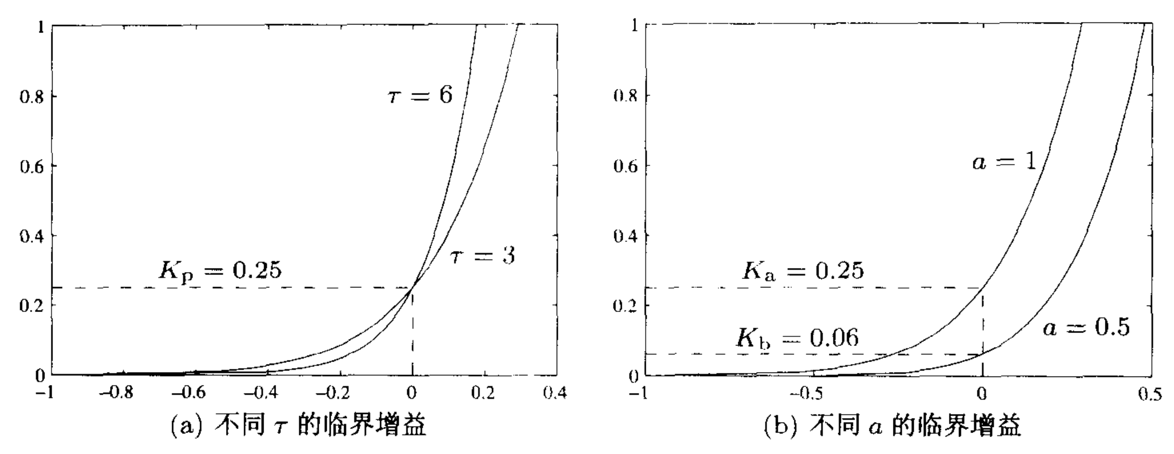
再考虑带有重极点的延迟开环模型 $G(s)=\dfrac{\mathrm{e}^{-\tau s}}{(s+a)^n}$，这样延迟微分方程可以写成 $(s+a)^n-K_\mathrm{p}\mathrm{e}^{-\tau s}=0$。令 $s_1=s+a$，则方程 $(s+a)^n-K_\mathrm{p}\mathrm{e}^{-\tau s}=0$ 可以变换成 $s_1\mathrm{e}^{\tau(s_1-a)/n}=\sqrt[n]{K_\mathrm{p}}$，这时，$\left(\dfrac{\tau}{n}s_1\right)\mathrm{e}^{\tau s_1/n}=\left(\dfrac{\tau}{n}\right)\mathrm{e}^{\tau a/n}\sqrt[n]{K_\mathrm{p}}$，该方程可以进一步简化为 $s_1=\dfrac{n}{\tau}\mathscr{W}\left(\dfrac{\tau}{n}\mathrm{e}^{\tau a/n}\sqrt[n]{K_\mathrm{p}}\right)$，考虑 $s=s_1-a$，为使得闭环系统稳定，应该有 $s<0$，即

$$\frac{n}{\tau}\mathscr{W}\left(\frac{\tau}{n}\mathrm{e}^{\tau a/n}\sqrt[n]{K_\mathrm{p}}\right)-a<0 \tag{5-1-1}$$

例 5-12 对不同的受控对象参数 (a,τ)，试求出闭环系统的稳定范围。

求解 先固定 $a=1$，选择两个不同的 τ 值 3, 6，则由下面的语句可以绘制出增益 K_p 和 s 之间的关系曲线，如图 5-6 (a) 所示。可见，这样得出的临界增益 $K_\mathrm{p}=0.25$，其值和 τ 无关。

```
>> n=2; tau=3; a=1; Kp=0:0.01:1;
   s_a=n*lambertw(tau*exp(tau*a/n)*Kp.^(1/n))/tau-a;
   tau=6; s_b=n*lambertw(tau*exp(tau*a/n)*Kp.^(1/n))/tau-a;
   plot(s_a,Kp, s_b,Kp)
```



(a) 不同 τ 的临界增益　　(b) 不同 a 的临界增益

图 5-6 基于 Lambert 函数的延迟微分方程稳定域研究

令 $\tau = 3$，并选择不同的 a 值，$a = 1, 0.5$，则由下面的语句可以绘制出增益 K_p 和 s 之间的关系曲线，如图 5-6 (b) 所示。可见，这样得出的临界增益将发生变化，使得 $K_a = 0.25$，$K_b = 0.06$。

```
>> n=2; tau=3; a=1; Kp=0:0.01:1;
   s_a=n*lambertw(tau*exp(tau*a/n)*Kp.^(1/n))/tau-a;
   a=0.5; s_b=n*lambertw(tau*exp(tau*a/n)*Kp.^(1/n))/tau-a;
   plot(s_a,Kp, s_b,Kp)
```

3. 实际方程求解举例 —— 代数 Riccati 方程求解

当然，代数 Riccati 方程数值求解有很好的线性代数算法，如基于 Schur 变换的算法[2] 和基于矩阵广义特征值的算法[3]，MATLAB 也有现成的求解函数 are()。这里以典型的代数 Riccati 方程为例，介绍在带有技巧的特殊求解方法未知时，如何用方程求解的底层方法求解已知方程，并应用此方法求解各种未被求解过的扩展 Riccati 方程。

考虑式 (3-4-17) 中给出的代数 Riccati 方程，重写如下

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} - \boldsymbol{X}\boldsymbol{B}\boldsymbol{X} + \boldsymbol{C} = \boldsymbol{0} \tag{5-1-2}$$

为编写求解程序通用起见，无需假定 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 矩阵为对称矩阵，这样方程共有 n^2 个参数需要求解，而由矩阵方程相应元素相等的方式正好可以列出 n^2 个方程，求解这些方程就能得出原 Riccati 方程的解。

根据上面的分析，方程中 $\boldsymbol{X}$ 矩阵的系数 x_{ij} 可以重新写成列向量 $\boldsymbol{x}$，亦即 $\boldsymbol{X}(:)$，方程的误差也可以将实际误差展成误差矩阵，这样的问题就可以描述成 $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ 的方程形式并由下面的 MATLAB 函数来描述

```
function y=my_riccati(x,A,B,C)
X=reshape(x,size(A)); y1=A'*X+X*A-X*B*X+C; y=y1(:);
```

由前面介绍的代数方程求解函数 fsolve() 可以编写出下面的代数 Riccati 方程求解函数，这里求解方程时采用了随机初值的方法。

```
function X=riccati_solve(A,B,C,x0)
if nargin==3, x0=rand(size(A)); end
x=fsolve('my_riccati',x0(:),[],A,B,C); X=reshape(x,size(A));
```

作为一种特殊情况，若假设 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 矩阵为对称矩阵，则 $\boldsymbol{X}$ 也是对称矩阵，这时需要求解的参数一共 $n(n+1)/2$ 个，引入一个列向量 $\boldsymbol{x}$，使得

$$x_i = X_{i,1}, i = 1, \cdots, n, x_{n+i} = X_{i,2}, i = 2, \cdots, n, \cdots \tag{5-1-3}$$

该规律可以写成

$$x_{(2n-j+2)(j-1)/2+i} = X_{i,j},\ j = 1, 2, \cdots, n, i = j, j+1, \cdots, n \tag{5-1-4}$$

用这样的方法看似可以大大减少方程的未知变量个数，但编程将变得很复杂，所以对规模不大的 Riccati 方程来说，无需专门为其编写求解函数，利用上面编写的函数就能直接求解。

例 5-13 试用这里给出的程序求解例 3-72 中给出的 Riccati 方程。

$$\boldsymbol{A} = \begin{bmatrix} -2 & 1 & -3 \\ -1 & 0 & -2 \\ 0 & -1 & -2 \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} 2 & 2 & -2 \\ -1 & 5 & -2 \\ -1 & 1 & 2 \end{bmatrix},\ \boldsymbol{C} = \begin{bmatrix} 5 & -4 & 4 \\ 1 & 0 & 4 \\ 1 & -1 & 5 \end{bmatrix}$$

求解 用下面的语句可以直接求解 Riccati 方程，并判定解的精度。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
   C=[5 -4 4; 1 0 4; 1 -1 5]; X=riccati_solve(A,B,C),
   norm(A'*X+X*A-X*B*X+C)
```

这样可以得出 Riccati 方程的解，并求出误差矩阵的范数为 1.0406×10^{-15}。

$$\boldsymbol{X} = \begin{bmatrix} -0.1538 & 0.10866 & 0.46226 \\ 2.0277 & -1.7437 & 1.3475 \\ 1.9003 & -1.7513 & 0.50571 \end{bmatrix}$$

与例 3-72 的结果对比可以立即发现，这两个结果是完全不同的，经检验这里得出的结果也满足原方程，所以原方程的解不惟一。其实这也是可以理解的，因为原来的方程是二次方程，所以有两个根是正常的。因为这里给出的方程解初值是随机的，所以反复调用几次本函数则可能得出例 3-72 中的解。

4. 扩展 Riccati 方程的求解 (I)

标准的 Riccati 方程要求一次项为 $\boldsymbol{AX} + \boldsymbol{XA}^{\mathrm{T}}$，仿照 Sylvester 方程，可以考虑将一次项修正为 $\boldsymbol{AX} + \boldsymbol{XD}$，这样扩展的 Riccati 方程可以定义为

$$\boldsymbol{AX} + \boldsymbol{XD} - \boldsymbol{XBX} + \boldsymbol{C} = \boldsymbol{0} \tag{5-1-5}$$

因为没有现成的算法求解该方程，所以可以仿照前面的方法，利用方程数值解法来直接搜索方程的数值解。例如编写出下面的求解函数

```
function X=e_riccati_solve(A,B,C,D,x0)
if nargin==4, x0=rand(size(A)); end
x=fsolve('my_e_riccati',x0(:),[],A,B,C,D); X=reshape(x,size(A));
```

这里求解方程时仍采用了随机初值的方法。另外，my_e_riccati() 函数是能够将扩展 Riccati 代数方程转换成一般向量型方程的 MATLAB 函数，可以如下编写出下面的求解函数

```
function y=my_e_riccati(x,A,B,C,D)
X=reshape(x,size(A)); y1=A*X+X*D-X*B*X+C; y=y1(:);
```

例 5-14 试求出扩展 Riccati 方程的全部根。

$$\boldsymbol{A}=\begin{bmatrix}2&1&9\\9&7&9\\6&5&3\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0&3&6\\8&2&0\\8&2&8\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}7&0&3\\5&6&4\\1&4&4\end{bmatrix},\ \boldsymbol{D}=\begin{bmatrix}3&9&5\\1&2&9\\3&3&0\end{bmatrix}$$

求解 由下面的语句可以求解这样扩展的 Riccati 方程。由于方程的求解中设置了随机初值，所以反复地调用该函数则可能得出不同的解。下面可以先输入系数矩阵，然后调用 e_riccati_solve() 函数，则能得出方程的一个解，再次调用则可能得出另一个解，如果不能得出，则反复调用几次就能得出方程全部的解。

```
>> A=[2,1,9; 9,7,9; 6,5,3]; B=[0,3,6; 8,2,0; 8,2,8];
   C=[7,0,3; 5,6,4; 1,4,4]; D=[3,9,5; 1,2,9; 3,3,0];
   x=e_riccati_solve(A,B,C,D)
```

可以得出下面的两个解

$$\boldsymbol{x}_1=\begin{bmatrix}0.78765&0.42984&-2.2827\\-0.60885&-0.26903&0.72972\\-0.65535&-0.34174&1.1487\end{bmatrix},\ \boldsymbol{x}_2=\begin{bmatrix}1.4567&1.3663&0.38581\\-0.28228&0.18807&2.0323\\-0.59091&-0.25153&1.4057\end{bmatrix}$$

5. 扩展 Riccati 方程的求解 (II)

对二次项稍加修改，则可以得出另一种扩展的 Riccati 方程

$$\boldsymbol{AX}+\boldsymbol{XD}-\boldsymbol{XBX}^{\mathrm{T}}+\boldsymbol{C}=\boldsymbol{0} \tag{5-1-6}$$

采用前面介绍的纯数值方法就可以得出该方程的解。

例 5-15 试求解扩展 Riccati 方程 (5-1-6)，其中 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 这四个矩阵和上例一致。

求解 仿照前面的例子可以写出下面的求解函数

```
function X=e_riccati_solve2(A,B,C,D,x0)
if nargin==4, x0=rand(size(A)); end
```

```
x=fsolve('my_e_riccati2',x0(:),[],A,B,C,D); X=reshape(x,size(A));
```

其中，描述这个特殊 Riccati 方程的 MATLAB 函数可以写成

```
function y=my_e_riccati2(x,A,B,C,D)
X=reshape(x,size(A)); y1=A*X+X*D-X*B*X.'+C; y=y1(:);
```

这时，通过下面函数可以对该方程求解

```
>> A=[2,1,9; 9,7,9; 6,5,3]; B=[0,3,6; 8,2,0; 8,2,8];
   C=[7,0,3; 5,6,4; 1,4,4]; D=[3,9,5; 1,2,9; 3,3,0];
   x=e_riccati_solve2(A,B,C,D)
```

反复调用该函数可以发现，可能得到下面三个解

$$\boldsymbol{x}_1=\begin{bmatrix}1.7539 & 1.2408 & -0.00023348\\ 2.2114 & 3.3662 & -0.72222\\ 0.86565 & 1.8109 & -0.26194\end{bmatrix},\quad x_2=\begin{bmatrix}6.74 & -0.36997 & 0.8394\\ -1.7679 & -0.25863 & 1.4835\\ 1.7761 & -0.39974 & 1.0043\end{bmatrix}$$

$$\boldsymbol{x}_3=\begin{bmatrix}-0.43386 & 0.40803 & 0.14075\\ 1.3621 & -2.5373 & 1.4561\\ -1.0243 & 0.97048 & -1.0438\end{bmatrix}$$

6. 基于方程求解的隐式微分方程求解方法

前面给出的 ode15i() 函数可以求取隐式微分方程的解。事实上，通过代数方程的数值求解，则可以将隐式微分方程变换成一阶显式微分方程组，这样用 ode45() 函数可以直接求解对应的隐式微分方程。

例 5-16 考虑例 4-20 中给出的隐式微分方程，试将其立即变换成显式微分方程，并求出该方程的数值解。

求解 可以仍然选定状态变量 $x_1=x, x_2=\dot{x}, x_3=y, x_4=\dot{y}$，这样仍然有 $\dot{x}_1=x_2, \dot{x}_3=x_4$。显然，并不可能像例 4-14 那样解析地求解方程，得出 $\dot{x}_2$ 和 $\dot{x}_4$ 的显式表达式，但可以讨论该方程组数值解的方法对每组状态变量 $\boldsymbol{x}$ 得出它们的值。

由原来给出的方程，假设 $p_1=\ddot{x}$. $p_2=\ddot{y}$，则可以将原方程改写成

$$\begin{cases} p_1\sin x_4+p_2^2+2x_1x_3-x_1p_1x_4=0\\ x_1p_1p_2+\cos p_2-3x_3x_2\mathrm{e}^{-x_1}=0\end{cases}$$

依据该方程可以得出如下的 MATLAB 语句，从而写出相应的 M-函数来描述微分方程如下:

```
function dy=c5mimpode(t,x)
dx=@(p,x)[p(1)*sin(x(4))+p(2)^2+2*x(1)*x(3)*exp(-x(2))-x(1)*...
      p(1)*x(4); x(1)*p(1)*p(2)+cos(p(2))-3*x(3)*x(2)*exp(-x(1))];
ff=optimset; dx1=fsolve(dx,x([1,3]),ff,x);
dy=[x(2); dx1(1); x(4); dx1(2)];
```

这里因为有中间步骤，所以不适合用 inline() 函数或匿名函数描述整个微分方程。进入该函数时，由状态变量 x 和新定义的 p_1,p_2 可以写出匿名函数，描述未知量 p_i 代入方程后两个方程的误差，而这里 x 是作为已知的附加变量给出的。微分方程求解程序每次调用这个原型函数时均求解一次关于 p_i 的代数方程，得出的结果 p_1,p_2 实际上就是 $\dot{x}_2,\dot{x}_4$，这样就可以构造出微分方程对应的状态变量的导数。在求解代数方程中使用了一个小技巧，即代数方程的初值选择 $p_1(0)=x_1,p_2(0)=x_3$，这样会使得代数方程收敛速度加快，精度增高。

建立起微分方程模型后，就可以通过下面的语句直接求解微分方程，并绘制出状态变量的时间曲线，和图 4-15 给出的完全一致。

```
>> [t,x]=ode15s(@c5mimpode,[0,2],[1,0,0,1]); plot(t,x)
```

5.2　无约束最优化问题求解

无约束最优化问题是最简单的一类最优化问题，其一般数学描述为

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{5-2-1}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$ 称为优化变量，$f(\cdot)$ 函数称为目标函数，该数学表示的含义亦即求取一组 $\boldsymbol{x}$ 向量，使得最优化目标函数 $f(\boldsymbol{x})$ 为最小，故这样的问题又称为最小化问题。其实，最小化是最优化问题的通用描述，它不失普遍性。如果要想求解最大化问题，那么只需给目标函数 $f(\boldsymbol{x})$ 乘一个负号就能立即将最大化问题转换成最小化问题。所以本书中描述的全部问题都是最小化问题。

本节先讨论无约束最优化问题的解析解法和图解法，然后侧重于介绍无约束最优化问题的 MATLAB 求解方法，并给出全局最优和局部最优的概念，并介绍如何用无约束最优化的方法求解最优控制器设计的问题。

5.2.1　解析解法和图解法

无约束最优化问题的最优点 $\boldsymbol{x}^*$ 处，目标函数 $f(\boldsymbol{x})$ 对 $\boldsymbol{x}$ 各个分量的一阶导数为 0，从而可以列出下面的方程：

$$\left.\frac{\partial f}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0,\ \left.\frac{\partial f}{\partial x_2}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0,\ \cdots,\ \left.\frac{\partial f}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0 \tag{5-2-2}$$

求解这些方程构成的联立方程可以得出极值点。其实，解出的一阶导数均为 0 的极值点不一定都是极小值的点，其中有的还可能是极大值点。极小值问题还应该有正的二阶导数。对于单变量的最优化问题，可以考虑采用解析解的方法进行求

解。然而多变量最优化问题因为需要将其转换成求解多元非线性方程，其难度也不低于直接求取最优化问题，所以没有必要采用解析解方法求解。

一元函数最优化问题的图解法也是很直观的，应绘制出该函数的曲线，在曲线上就能看出其最优值点。二元函数的最优化也可以通过图解法求出。但三元或多元函数，由于用图形没有办法表示，所以不适合用图解法求解。

例 5-17 例 5-1 中给出的方程 $f(t)=\mathrm{e}^{-3t}\sin(4t+2)+4\mathrm{e}^{-0.5t}\cos(2t)-0.5$，试用解析求解和图形求解的方法研究该函数的最优性。

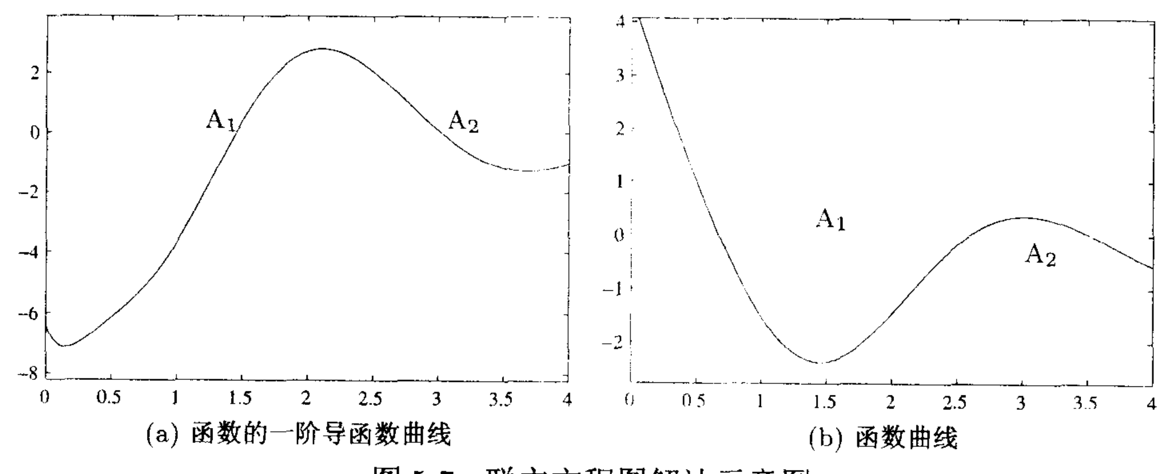
求解 可以先表示该函数，并解析地求解该函数的一阶导数，用 ezplot() 函数可以绘制出 $t\in[0,4]$ 区间内一阶导函数的曲线，如图 5-7 (a) 所示。

```
>> syms t; y=exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5;
   y1=diff(y,t); % 求取一阶导函数
   ezplot(y1,[0,4])  % 绘制出选定区间内一阶导函数曲线
```

其实，求解导函数等于 0 的方程不比直接求解其最优值简单。用图解法可以看出，在这个区间内有两个点，A_1 和 A_2，使得它们的一阶导函数为 0，但从其一阶导数走向看，A_2 点对应负的二阶导数值，所以该点对应于极大值点，而 A_1 点对应于正的二阶导数值，故为极小值点。A_1 点的值可以由下面的语句直接解出。

```
>> t0=solve(y1), ezplot(y,[0,4])     % 求出一阶导数等于零的点
   y2=diff(y1); b=subs(y2,t,t0)      % 并验证二阶导数为正
```

可以得出 $t_0=1.45284, b=7.8553>0$。这样，就可以求出函数的最小值。还可以用图形绘制的方法进一步验证得出的结果，如图 5-7 (b) 所示，可见，A_1 为最小值，A_2 为最大值。



(a) 函数的一阶导函数曲线　(b) 函数曲线

图 5-7 联立方程图解法示意图

然而因为给定的函数是非线性函数，所以用解析法或类似的方法求解最小值问题一点都不比直接求解最优化问题简单。因此，除演示之外，不建议用这样的方法求解该问题，而是直接采用最优化问题求解程序得出问题的解。

5.2.2 基于 MATLAB 的数值解法

MATLAB 语言中提供了求解无约束最优化的函数 fminsearch()，其最优化工具箱中还提供了函数 fminunc()，二者的调用格式完全一致，为

```
x=fminunc(Fun,x0)                                  % 最简求解语句
[x,f,flag,out]=fminunc(Fun,x0,opt,p1,p2,···)       % 一般求解格式
```

其输入与返回参数的定义与 fsolve() 函数中的控制变量完全一致。该函数主要采用了文献 [4] 中提出的单纯形算法。下面将通过例子来演示无约束最优化问题的数值解法。

例 5-18 已知二元函数 $z=f(x,y)=(x^2-2x)e^{-x^2-y^2-xy}$，试用 MATLAB 提供的求解函数求出其最小值，并用图形方法表示其求解过程。

求解 因为函数中给出的自变量是 x,y，而最优化函数需要求取的是自变量向量 $\boldsymbol{x}$，故在求解前应该先进行变量替换，如令 $x_1=x, x_2=y$，这样就可以用下面的语句由匿名函数形式定义出目标函数 f，然后将求解控制变量中的 Display 属性设置为 'iter'，这样可以显示中间的搜索结果。用下面的语句求解出最优解。

```
>> f=@(x)(x(1)^2-2*x(1))*exp(-x(1)^2-x(2)^2-x(1)*x(2));
   x0=[0; 0]; g=optimset; g.Display='iter'; x=fminsearch(f,x0,g)
```

经过 72 步迭代，137 次目标函数求值，得出 $\boldsymbol{x}=[0.6110537,-0.305578]^{\mathrm{T}}$。

同样的问题用 fminunc() 函数求解，则

```
>> x=fminunc(f,[0;.0],ff)
```

求解需要 7 次迭代，27 次目标函数的调用，得出的解为 $\boldsymbol{x}=[0.6110462,-0.3055242]^{\mathrm{T}}$。

比较两种方法，显然可以看出，用 fminunc() 函数的效率明显高于 fminsearch()。所以在无约束最优化问题求解时，如果安装了最优化工具箱则建议使用 fminunc() 函数。

在求解过程中，如果手工修改 fminunc() 下级的 fminusub() 函数文件，就可以追踪出各个搜索中间点的坐标。下面在图形上显示出搜索中间过程。假设选择 $\boldsymbol{x}_0=[2,1]^{\mathrm{T}}$，则由下面的语句可以得出所需的解，同时还能由修改的函数得出中间点坐标为

x_i	2	0.2401	−0.1398	0.2168	0.3355	0.5514	0.6129	0.6111
y_i	1	1.0502	0.5752	1.0210	−0.5508	−0.1775	−0.3053	−0.3058

用下面的语句可以绘制出搜索过程中间点的轨线，如图 5-8 所示。

```
>> xx=[2  0.2401 -0.1398 0.2168  0.3355  0.5514  0.6129  0.6111
       1  1.0502  0.5752 1.0210 -0.5508 -0.1775 -0.3053 -0.3058];
   [x,y]=meshgrid(-3:.1:3, -2:.1:2);
```

```
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
contour(x,y,z,30); line(xx(1,:),xx(2,:))
h=line(xx(1,:),xx(2,:)); set(h,'Marker','o')
```

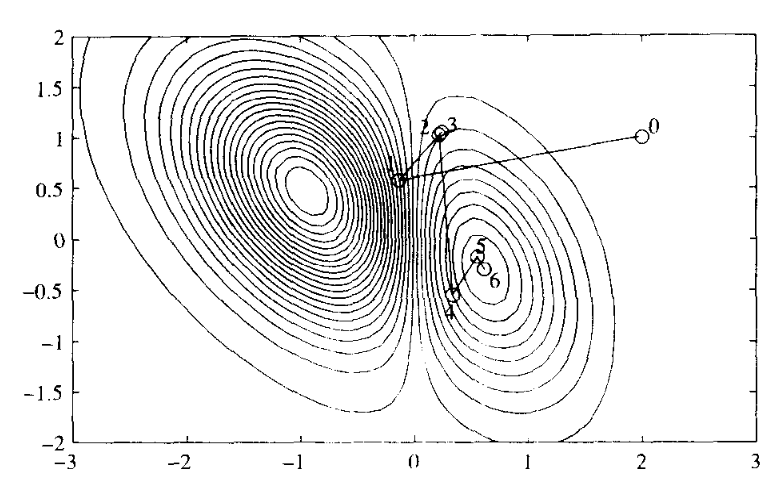


图 5-8 搜索过程中间点轨线示意图

5.2.3 全局最优解与局部最优解

以单变量 x 为例，无约束最优化问题函数有解的必要条件是 $\mathrm{d}f(x)/\mathrm{d}x=0$，但满足该条件的 x 值可能不惟一，可能存在多个解。从最优化搜索的角度来说，可能找到其中一个点。下面将通过例子引入全局最优解和局部最优解的概念。

例 5-19 假设目标函数为 $y(t)=\mathrm{e}^{-2t}\cos 10t+\mathrm{e}^{-3t-6}\sin 2t$，$t\geqslant 0$，试观察不同的初值能得出的最小值，并讨论局部最小值与全局最小值的概念。

求解 由给定的目标函数，可以立即写出用于无约束最优化搜索的 MATLAB 表示，采用下面的匿名函数或编写相应的 MATLAB 文件。

```
>> f=@(t)exp(-2*t).*cos(10*t)+exp(-3*(t+2)).*sin(2*t);
```

若选定初始搜索点为 $t_0=1$，则可以通过如下的语句获得目标函数的最优解

```
>> t0=1; [t1,f1]=fminsearch(f,t0)
```

得出的结果为 $t_1=0.92275390625, f_1=-0.15473299821860$。

若选择另一个初始点，则得出如下的最优解

```
>> t0=0.1; [t2,f2]=fminsearch(f,t0)
```

得出的结果为 $t_2=0.294453125, f_2=-0.54362$。

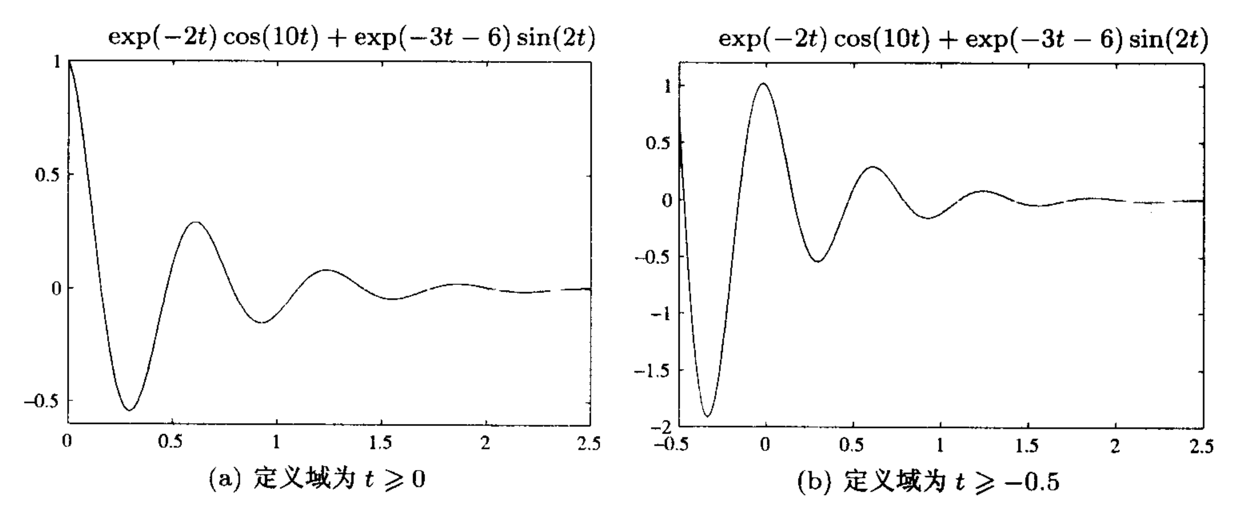
可见，给出不同的初值，此函数能得出不同的“最优解”，但从最优解处的函数值看，t_1 处的函数值显然大于 t_2 处的函数值。故可以得出结论，t_1 得出的并非真正的最优解，而是某种局部最优解。试凑其他的初值，如 $t_0=1.5,2,2.5,\cdots$ 还可能得出其他的局部最优值，这里不一一列出。用 MATLAB 的 ezplot() 函数可以绘制给定目标

函数 $y(t)$ 在 $t \in (0, 2.5)$ 定义域内的曲线，如图 5-9 (a) 所示，区间内的全局和局部最优值均由虚线表示出来。

```
>> syms t; y=exp(-2*t)*cos(10*t)+exp(-3*(t+2))*sin(2*t);
   ezplot(y,[0,2.5]); ylim([-0.6,1])
```

从图 5-9 (a) 可以看出，在 $t \geqslant 0$ 定义域内，t_2 点是目标函数真正的最优值，在最优化理论中又称为全局最优解，由于初值选择不同的值可能得出不同的最优值，其中有些是局部最优值，所以这里给出的 fminsearch() 函数并不能保证求出全局最优值。事实上，目前所有的最优化算法没有哪一种能保证能求出最优化问题的全局最优解。

现在再考虑更大些的定义域，即 $t \geqslant -0.5$，则用下面的语句能绘制出该函数在新定义域内的曲线，如图 5-9 (b) 所示。



(a) 定义域为 $t \geqslant 0$　　(b) 定义域为 $t \geqslant -0.5$

图 5-9　全局最优解与局部最优解

```
>> ezplot(y,[-0.5,2.5]); ylim([-2,1.2])
   t0=-0.2; [t3,f3]=fminsearch(f,t0);
```

可以得出方程的解为 $t_3 = -0.334, f_3 = -1.9163$。从图 5-9 (b) 可见，前面得出的全局最优解在新的定义域内就不再是全局最优解了，比起新的最优解来说，它只是局部最优解，若进一步扩大定义域，则得出的 t_3 将也不再是全局最优解了，而称为局部最优解。若将定义域扩展到 $t \in (-\infty, \infty)$ 区间，则原函数将没有真正意义的全局最优解。

通过上面的例子可以看出，利用 MATLAB 的求解函数或其他现成的最优化问题求解函数可能得出局部最优解，而不是全局最优解，这就需要读者自己去试取不同的初值，看看得出的最优解是否相同，如果不同，则比较一下哪个是局部最优解。遗传算法提供了一种同时试测不同初值的算法，在求解全局最优解上有一定的改进，但也不能保证得出全局最优解。有关遗传算法及其在最优化问题中的应用请参见 7.3 节。

5.2.4 利用梯度求解最优化问题

有时最优化问题求解速度较慢，有时甚至无法搜索到较精确的最优点，尤其是变量较多的最优化问题，所以需要引入目标函数梯度，以加快计算速度，改进搜索精度。然而，有时计算梯度也是需要时间的，也会影响整个运算速度，所以实际求解时应该考虑是不是值得引入梯度的概念。

在利用 MATLAB 最优化工具箱求解最优化问题时，也应该和目标函数在同一函数中描述梯度函数，亦即这时 MATLAB 的目标函数应该返回两个变量，第一个变量仍然表示目标函数，第二个变量可以返回梯度函数。同时，还应该将求解控制变量的 `GradObj` 属性设置成 `'on'`，这样就可以利用已知的梯度信息来求解最优化问题了。

例 5-20 试求解 Rosenbrock 函数[5] $f(x_1,x_2)=100(x_2-x_1^2)^2+(1-x_1)^2$ 的无约束最优化问题。

求解 从目标函数可以看出，由于它为两个平方数的和，所以当 $x_2=x_1=1$ 时，整个目标函数有最小值 0。下面语句可以绘制出目标函数的三维等高线图，如图 5-10 所示。

```
>> [x,y]=meshgrid(0.5:0.01:1.5); z=100*(y.^2-x).^2+(1-x).^2;
   contour3(x,y,z,100), zlim([0,310])
```

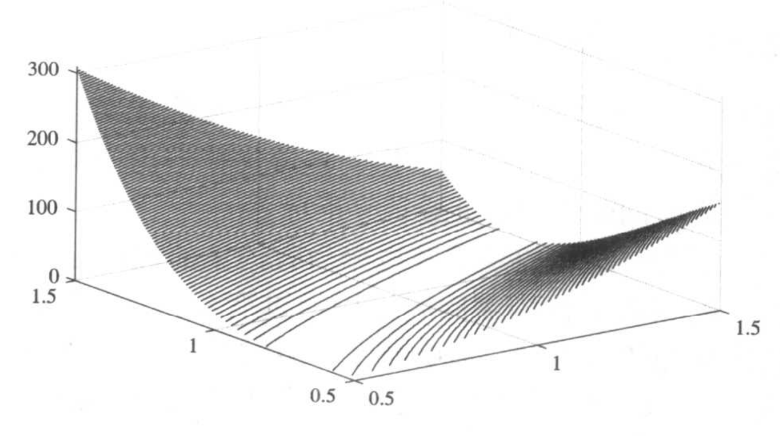


图 5-10 Rosenbrock 目标函数的三维等高线图

由得出的曲线看，其最小值点在图中的一个很窄的白色带状区域内，故 Rosenbrock 目标函数又称为香蕉形函数，而在这个区域内的函数值变化较平缓，这就给最优化求值带来很多麻烦。该函数经常用来测试最优化算法的优劣。现在观察用下面语句求解最优化问题。

```
>> f=@(x)[100*(x(2)-x(1)^2)^2+(1-x(1))^2];
   ff=optimset; ff.TolX=1e-10; ff.TolFun=1e-20; ff.Display='iter';
   x=fminunc(f,[0;0],ff)
```

经过 20 步迭代，80 次目标函数求值，得出的解为 $x=[0.999995588,0.999991167]^{\mathrm{T}}$。

可见，该算法无法在给定的步数内精确搜索到真值 $(1,1)$，用传统的最速下降法更无法搜索到真值。这时需要引入梯度的概念。

对给定的 Rosenbrock 函数，利用符号运算工具箱即可以求出其梯度向量为

```
>> syms x1 x2; f=100*(x2-x1^2)^2+(1-x1)^2; J=jacobian(f,[x1,x2])
```

可以求出该函数的 Jacobi 矩阵为 $\begin{bmatrix} -400(x_2-x_1^2)x_1-2+2x_1 \\ 200x_2-200x_1^2 \end{bmatrix}$。这时，可以在目标函数中描述其梯度，故需要重新编写目标函数为

```
function [y,Gy]=c5fun3(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
Gy=[-400*(x(2)-x(1)^2)*x(1)-2+2*x(1); 200*x(2)-200*x(1)^2];
```

这样，就应该给出如下命令来求解最优化问题。

```
>> ff.GradObj='on'; x=fminunc(@c5fun3,[0;0],ff)
```

得出的解为 $\boldsymbol{x}=[0.99999999999999, 0.99999999999997]^{\mathrm{T}}$，求解过程为 18 次迭代。可见，引入了梯度则可以明显加快搜索的进度，且最优解也基本上逼近于真值，这是不使用梯度不可能得到的，所以从本例可以看出梯度在搜索中的作用。然而，在有些例子中引入梯度也不是很必要，因为梯度本身的计算和编程需要更多的时间。

5.2.5 利用最优化方法设计最优控制器

所谓“最优控制”，就是在一定的具体条件下，要完成某个控制任务，使得选定指标最小或最大的控制，这里所谓指标就是最优化理论中的目标函数，常用的目标函数可以设置成时间最短、能量最省等指标。

和最优化技术类似，最优控制问题也分为有约束的最优控制问题和无约束的最优控制问题。无约束的最优控制问题可以通过变分法[6,7]来求解，对于小规模问题，可能求解出问题的解析解，例如前面介绍过的二次型最优控制器设计问题就有直接求解公式。有约束的最优化问题则较难处理，需要借助于 Pontryagin 的极大值原理。

针对具体的最优控制问题，应该选择什么样的指标，使得控制的效果达到最好，这一直是控制理论界学者与工程技术人员感兴趣的问题。早期的控制研究因为缺乏切实可行的计算机求解工具，所以研究者更侧重于获得可以得出解析解的问题描述。例如，对线性状态空间模型的最优控制研究，首先引入了具有二次型性能指标的控制目标，并推导出了基于微分 Riccati 方程的最优控制的解，后来又因为这样的微分方程难于求解，所以化简了问题的描述，将二次型最优控制问题简化成无限时间的调节问题，将求解的方程简化成代数 Riccati 方程，使得最优调节器的设计成为可能。但这样难免带来新的问题：首先将控制问题简化成调节问题，从趋势上能得出好的调节器，但对于动态性能的因素就完全忽略了，这使得系

统求解的问题发生了变化。另外，通过人为引入加权矩阵 $\boldsymbol{Q},\boldsymbol{R}$，这样虽然能得出数学上较漂亮的状态反馈规律，但这两个加权矩阵却至今没有被广泛认可的选择方法，这使得系统的最优准则带有一定的人为因素，没有足够的客观性。

那么真正的受控对象到底需要什么样的"最优"控制器呢？不同的控制问题，不同的控制学者和工程技术人员可能有不同的观点。现在以跟踪控制问题为例，显然跟踪问题最令人感兴趣的指标是让跟踪误差尽可能小，所以可以围绕跟踪误差设计最优指标。假设跟踪误差为 $e(t)$，则可以定义出 ISE 指标和 ITAE 指标

$$J_{\rm ISE}=\int_0^\infty e^2(t){\rm d}t,\ \ J_{\rm ITAE}=\int_0^\infty t|e(t)|{\rm d}t \tag{5-2-3}$$

随着像 MATLAB 这样强有力的计算机语言与工具普及起来之后，很多最优控制问题可以变换成一般的最优化问题，用数值最优化方法就可以简单地求解。这样的求解虽然没有完美的数学形式，但有时还是很实用的。下面通过例子演示依赖纯数值方法最优控制器的设计与应用。

例 5-21 考虑受控对象为 $G(s)=\dfrac{1}{(s+1)^5}$，试为其设计出一个具有最小 ISE 准则的最优控制器。

求解 可以推导出系统的误差信号的 Laplace 变换式 $E(s)$，这样 ISE 目标函数可以改为频域求解方法直接求解

$$J=\frac{1}{2\pi{\rm j}}\int_{-{\rm j}\omega}^{{\rm j}\omega}E(s)E(-s){\rm d}s \tag{5-2-4}$$

而该复数积分在文献 [8] 中有递推算法和程序直接求解。对状态方程模型来说，还可以用相应 Lyapunov 方程的求解方法解出误差 ISE 准则的解析解，或采用 $\mathcal{H}_2$ 范数的形式求出，MATLAB 的 norm() 函数也可以求解该值。这里演示由 norm() 函数求取最优控制器参数的方法。若系统不稳定，则 norm() 函数将返回 Inf，使得目标函数的计算出现问题，所以这时可以人为地令目标函数的值等于 1000。根据前面的分析，可以写出如下的 MATLAB 函数描述目标函数

```
function e=c5mopid(x,G,s)
E=minreal(feedback(1,G*(x(1)+x(2)/s+x(3)*s))/s); e=norm(E);
if ~isfinite(e), e=1000; end
```

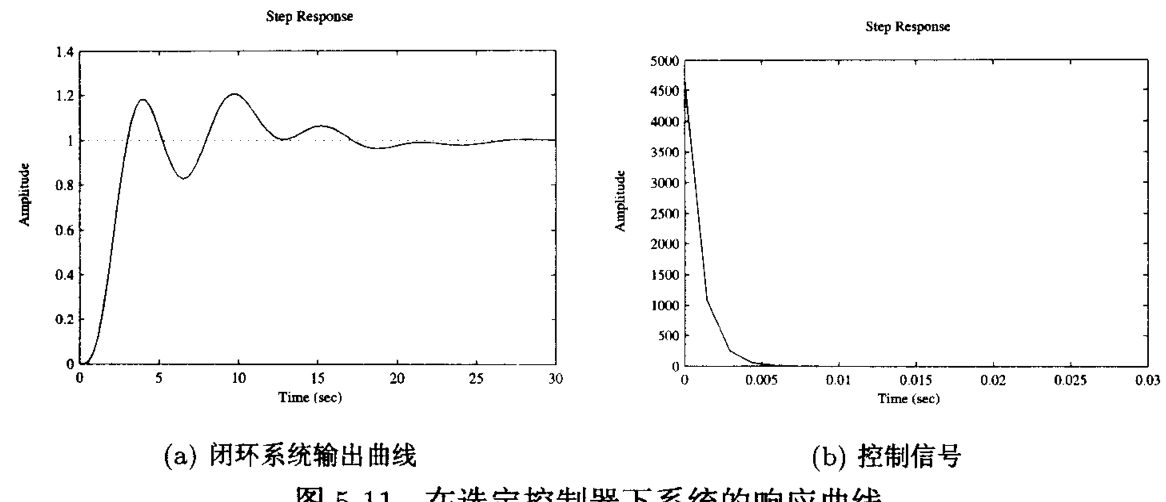
其中，优化变量 $\boldsymbol{x}=[K_{\rm p},K_{\rm i},K_{\rm d}]$。由下面语句即可以搜索出使得 ISE 准则最小的控制器参数

```
>> s=tf('s'); G=1/(s+1)^5; x0=[0.5 0.1 0];
   x=fminunc(@c5mopid,x0,[],G,s)
```

得出的结果为 $\boldsymbol{x}=[1.5375,0.6908,4.6574]^{\rm T}$，亦即 PID 控制器参数为 $K_{\rm p}=1.5375,K_{\rm i}=0.6908,K_{\rm d}=4.6574$。在此控制器的作用下，闭环系统的输出曲线如图 5-11 (a) 所示，

控制信号如图 5-11 (b) 所示。可以看出，输入信号在初始时刻过大。

```
>> Gc=x(1)+x(2)/s+x(3)*s/(0.001*s+1); step(feedback(G*Gc,1))
   figure; step(feedback(Gc,G))
```



(a) 闭环系统输出曲线　　(b) 控制信号

图 5-11　在选定控制器下系统的响应曲线

由前面的设计例子可见，可以通过引入实用的目标函数，然后借助现代最新计算机数学语言求解相应的最优化问题，得出更有意义的最优控制器。然而，由前面的最优控制器设计结果，自然会引出下面几个问题:

① **最优指标选择是否合理**: 从得出的控制效果可见，输出信号是基本上令人满意，但是不是存在更有意义的最优性指标? 如何选择目标函数能得出更好的结果?

② **如何处理系统中的非线性**: 前面例子中计算目标函数是由求 Laplace 变换表达式的 $\mathcal{H}_2$ 范数而实现的，而该方法显然不适合于非线性系统的最优控制器设计，如果系统中含有非线性环节，应该如何处理?

③ **控制信号过大不可实现怎么办**: 由前面例子中可见，得出的控制器信号在 t 很微小时可能过大，实际系统中不能接受这样大的控制信号，所以在实际控制中应该让控制信号经过一个饱和装置，以确保该信号不超过容许的范围。这样做显然需要在系统中人为引入非线性现象。

综上所述，如果出现上面任意一种情况，用 $\mathcal{H}_2$ 范数计算目标函数都是不可行的。所以应该探索和使用其他的最优控制器设计方法，比如下面要着重介绍的 MATLAB/Simulink 结合的设计方法。

例 5-22 仍考虑前面的例子，如果使用 ITAE 准则，且 PID 控制器输出的信号满足 $|u(t)| \leqslant 3$，试设计出相应的控制器。

求解 和 ISE 指标对应于 $\mathcal{H}_2$ 范数不同，ITAE 指标没有解析解方法，只能采用仿真方法求解，然而不可能仿真到 $t\to\infty$ 时刻，只能进行有限时刻的仿真，这样就涉及到仿真终止时刻 $t_{\rm f}$ 的选择了，后面将探讨选择的问题。

现在考虑控制器信号的限制问题，当然不可能指望用解析的方法保证其不超过限制，最简单的解决方法是在控制器输出端加一个饱和非线性环节，一旦计算出来的控制信号超过限制，则取饱和值。

根据前面的分析，可以搭建起如图 5-12 所示的 PID 控制的 Simulink 模型，在模

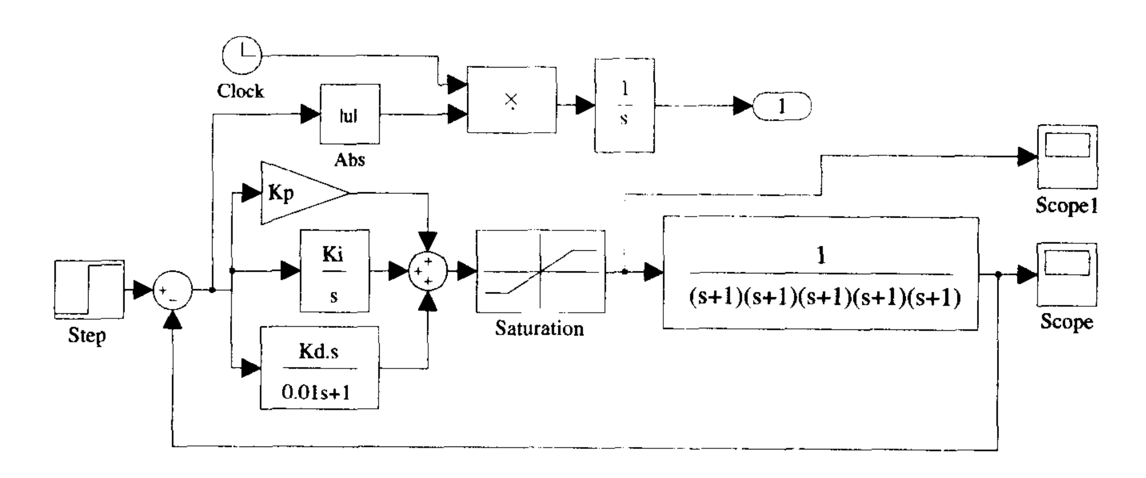


图 5-12 PID 控制的 Simulink 模型 (文件名：c5moptpid.mdl)

型中除了常规的 PID 控制系统之外，还搭建起 ITAE 积分，其在仿真终止时刻 $t_{\rm f}$ 的值可以定义为目标函数，饱和非线性确保了控制信号不会过大。根据这个模型可以编写出下面的 MATLAB 函数来描述目标函数

```
function y=c5optfun(x)
assignin('base','Kp',x(1)); assignin('base','Ki',x(2));
assignin('base','Kd',x(3));
[t,x1,y1]=sim('c5moptpid.mdl',[0,30]); y=y1(end);
```

定义了目标函数，则可以由下面的语句得出最优的 PID 控制器，输出信号的曲线如图 5-13 (a) 所示，其控制效果明显优于基于 ISE 目标函数的控制器效果。同时，控制信号如图 5-13 (b) 所示，该信号被严格控制到指定的范围内。

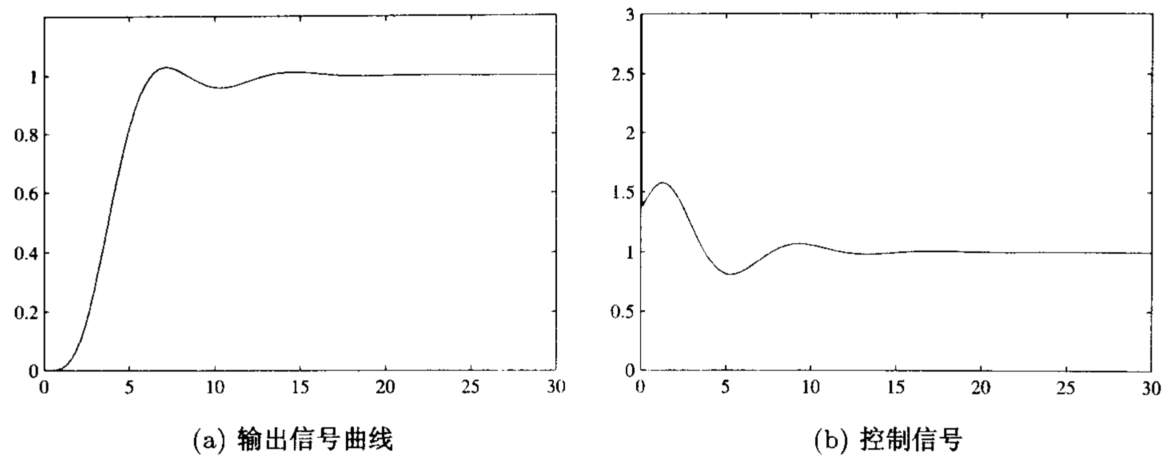
```
>> x0=[0.5 0.1 0]; x=fminunc(@c5optfun,x0)
```

可以通过寻优过程得出最优向量 $\boldsymbol{x}=[1.3504, 0.2645, 1.8667]^{\rm T}$，亦即 PID 控制器参数为 $K_{\rm p}=1.3504, K_{\rm i}=0.2645, K_{\rm d}=1.8667$。

现在考虑不同的 $t_{\rm f}$ 值对最优控制效果的影响，可以将目标函数修改为

```
function y=c5optfun1(x,tf)
assignin('base','Kp',x(1)); assignin('base','Ki',x(2));
assignin('base','Kd',x(3));
[t,x1,y1]=sim('c5moptpid.mdl',[0,tf]); y=y1(end);
```

对不同的 $t_{\rm f}$ 选值，对应的控制器参数和 ITAE 积分值可以由下面的语句得出，如表 5-2 所示。可见，在合理的范围内，$t_{\rm f}$ 的选择对控制器设计影响不大。



(a) 输出信号曲线　　(b) 控制信号

图 5-13 在新控制器下系统的响应曲线

```
>> tf_vec=[15:5:60]; X=[];
   for tf=tf_vec,
      [x,fopt]=fminunc(@c5optfun1,x,[],tf); X=[X; tf, x(:).' fopt];
   end
```

表 5-2 不同 t_f 对控制效果的比较

t_f	K_p	K_i	K_d	ITAE	t_f	K_p	K_i	K_d	ITAE
15	1.3602	0.27626	1.9773	9.3207	20	1.4235	0.27228	2.0731	9.7011
25	1.3524	0.26555	1.8846	9.945	30	1.3506	0.26454	1.8669	9.9814
35	1.3488	0.26395	1.859	10.008	40	1.3457	0.26363	1.8495	10.016
45	1.346	0.26357	1.8493	10.019	50	1.3452	0.26351	1.8472	10.021
55	1.345	0.26349	1.8467	10.021	60	1.345	0.26349	1.8467	10.021

前面分别求出了 ITAE 准则和 ISE 准则下的最优控制器并比较了控制效果，从动态角度看，ITAE 准则远远优于 ISE 准则的初步结论。因为 ISE 准则对任意时刻的误差信号都同等地对待，而 ITAE 准则对 t 较大的误差有所侧重，这就是为什么在 ITAE 准则下系统输出能尽快地到达稳态值的真正原因。那为什么在传统文献中 ISE 的应用更广泛呢？主要是因为线性系统在 ISE 准则或 $\mathcal{H}_2$ 准则下容易得出解析解。现在有了强大的计算机和专用软件，ITAE 这种能得出更客观最优控制效果的方法应该能更广泛地应用。

现在解释一下什么是"合理"范围。由于 ITAE 积分的被积函数 $t|e(t)|$ 是非负的，所以 ITAE 积分是递增的，而 ITAE 积分值增大到某个值时基本就不增加了，因为这时 $e(t)\to 0$。常规的 ITAE 积分曲线如图 5-14 给出。在得出这样的曲线后，用户可以从中找出一个能使得该曲线平稳的点作为 t_f 值的候选点。如果 t_f

选得过大，例如比平稳处 t 值大很多倍，则这时系统暂态处的动态性能可能被忽略。所以 t_f 应该选择为平稳处 1～2 倍比较理想。

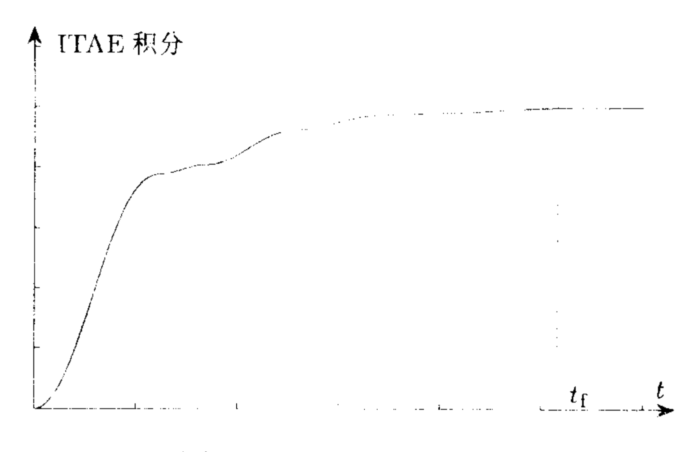


图 5-14 ITAE 积分曲线

5.2.6 带有变量边界约束的最优化问题求解

前面介绍的最优化问题是纯粹的无约束最优化问题，在实际应用中，更常见的无约束最优化问题并不完全是绝对的无约束问题，通常优化变量需要在指定的范围内选择，所以这样的问题一般可以表示成

$$\min_{\boldsymbol{x} \text{ s.t. } \boldsymbol{x}_\mathrm{m} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_\mathrm{M}} f(\boldsymbol{x}) \tag{5-2-5}$$

其中，记号 s.t. 是英文 subject to 的缩写，表示满足后面的关系。所以本式所描述的问题是，$\boldsymbol{x}$ 在指定的范围内取多少时能使得目标函数取最优值。比如在前面的 PID 控制器设计中，可以人为地令 PID 参数均大于零，这样就可以用 $\boldsymbol{x}_\mathrm{m} = [0,0,0]^\mathrm{T}$ 来表示约束。这样的问题由 `fminsearch()` 函数是不能直接求解的。John D'Errico 开发的 `fminsearchbnd()` 函数扩展了现有函数的功能，能直接求解这样的问题[❶]，该函数的调用格式为

```
x=fminsearchbnd(Fun,x0,xm,xM)
[x,f,flag,out]=fminsearchbnd(Fun,x0, xm,xM,opt,p1,p2,···)
```

如果上界或下界约束没有给出，则可以将其设置为空矩阵 []。

例 5-23 重新考虑例 5-20 中研究的 Rosenbrock 函数的最优化问题。显然，$x_1 = x_2 = 1$ 是该问题的最优解。如果 x_1 和 x_2 的允许区间不是整个区间，而只允许 $x_1 \in (2,4)$，$x_2 \in (3,6)$，试得出满足要求的最优解。

---

❶ 地址：http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8277&objectType=file。

求解 根据 x_1, x_2 的范围，可以直接得出 $\boldsymbol{x}_{\rm m} = [2,3]^{\rm T}, \boldsymbol{x}_{\rm M} = [4,6]^{\rm T}$，这样调用 fminsearchbnd() 函数则可以得出在容许范围内的最优解

```
>> f=@(x)[100*(x(2)-x(1)^2)^2+(1-x(1))^2]; xm=[2,3]; xM=[4,6];
   x=fminsearchbnd(f,[0,0],xm,xM)
```

得出的解为 $\boldsymbol{x} = [2, 3.9999996]^{\rm T}$。

现在考虑线性系统 $\mathcal{H}_\infty$ 范数的计算问题。第 3 章曾经介绍过，如果线性系统由状态方程 $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ 给出，则 $\mathcal{H}_\infty$ 范数可以由现成函数 norm() 计算出来，该算法采用二分法求解相应的最优化问题。现在以该问题为背景介绍基于无约束最优化技术在范数计算中的应用。计算 $\mathcal{H}_\infty$ 范数基于下述的定理[9]：

给定一个正数 $\gamma > 0$，当且仅当 Hamilton 矩阵

$$\boldsymbol{H}_\gamma = \begin{bmatrix} \boldsymbol{A} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{D}^{\rm T}\boldsymbol{C} & -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\rm T} \\ \boldsymbol{C}^{\rm T}(\boldsymbol{I} + \boldsymbol{D}\boldsymbol{R}^{-1}\boldsymbol{D}^{\rm T})\boldsymbol{C} & -(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{D}^{\rm T}\boldsymbol{C})^{\rm T} \end{bmatrix} \tag{5-2-6}$$

没有纯虚数的特征值，则不等式 $\|\boldsymbol{G}\|_\infty < \gamma$ 成立，其中 $\boldsymbol{R} = \gamma^2\boldsymbol{I} - \boldsymbol{D}^{\rm T}\boldsymbol{D} > 0$。

根据这个定理就可以通过最优化的方法求出最大的 γ 值，也就是最小的 $-\gamma$ 值。假设求解变量 γ 为输入变量，$-\gamma$ 为目标函数，$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ 为附加变量。若上面的不等式不成立，则可以人为地将目标函数设置为 100，这样可以编写出下面的目标函数

```
function y=hinf_ineq(gam,A,B,C,D)
y=-gam; I=eye(size(D*D')); R=gam^2*I-D'*D; iR=inv(R);
H=[A+B*iR*D'*C, -B*iR*B'; C'*(I+D*iR*D')*C, -(A+B*iR*D'*C)'];
if all(abs(real(eig(H)))>1e-8) | any(real(eig(R))<=0), y=100; end
```

由于前提是 $\gamma > 0$，所以特别需要 fminsearchbnd() 函数来求解问题。

例 5-24 假设系统的模型如下，试求出其 $\mathcal{H}_\infty$ 范数。

$$\boldsymbol{A} = \begin{bmatrix} -4 & -3 & 0 & -1 \\ -3 & -7 & 0 & -3 \\ 0 & 0 & -13 & -1 \\ -1 & -3 & -1 & -10 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 0 \\ -4 \\ 2 \\ 5 \end{bmatrix}, \ \boldsymbol{C} = [\,0,\ 0,\ 4,\ 0\,], \ \boldsymbol{D} = 0$$

求解 调用 norm() 函数，可以得出 $\mathcal{H}_\infty$ 范数为 0.4639。调用最优化求解的方法也能得出同样的结果。注意，初值应该选择一个尽可能小的正数。

```
>> A=[-4,-3,0,-1; -3,-7,0,-3; 0,0,-13,-1; -1,-3,-1,-10];
   B=[0; -4; 2; 5]; C=[0,0,4,0]; D=0; norm(ss(A,B,C,D))
   x=fminsearchbnd(@hinf_ineq,1e-5,0,100,[],A,B,C,D)
```

5.3 有约束最优化问题的计算机求解

有约束最优化问题的一般描述为

$$\min_{\boldsymbol{x}\ \text{s.t.}\ \boldsymbol{G}(\boldsymbol{x})\leqslant 0} f(\boldsymbol{x}) \tag{5-3-1}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$。该数学表示的含义为求取一组 $\boldsymbol{x}$ 向量，使得在满足约束条件 $\boldsymbol{G}(\boldsymbol{x})\leqslant 0$ 的前提下能够使目标函数 $f(\boldsymbol{x})$ 最小化。在实际遇到的最优化问题中，有时约束条件可能是很复杂的，它既可以是等式约束，也可以是不等式约束，既可以是线性的，也可能是非线性的，有时甚至是不能用纯数学函数来描述。

本节首先给出可行解区域的概念，然后介绍几种成型的有约束最优化问题的求解方法，包括线性规划问题求解方法、二次型规划问题求解方法和一般非线性规划问题的求解方法，并介绍有约束最优化问题在最优控制器设计中的应用。

5.3.1 约束条件与可行解区域

满足约束条件 $\boldsymbol{G}(\boldsymbol{x})\leqslant \boldsymbol{0}$ 的 $\boldsymbol{x}$ 范围称为可行解区域 (feasible region)。下面通过例子演示二元问题的可行解范围与图解结果。

例 5-25 考虑下面二元最优化问题的求解，试用图解方法对该问题进行研究。

$$\max_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}9\geqslant x_1^2+x_2^2\\ x_1+x_2\leqslant 1\end{array}\right.} (-x_1^2-x_2)$$

求解 由约束条件可见，若在 $[-3,3]$ 区间选择网格，则可以得出无约束时目标函数的三维图形数据。可以用下面的语句获得这些数据。

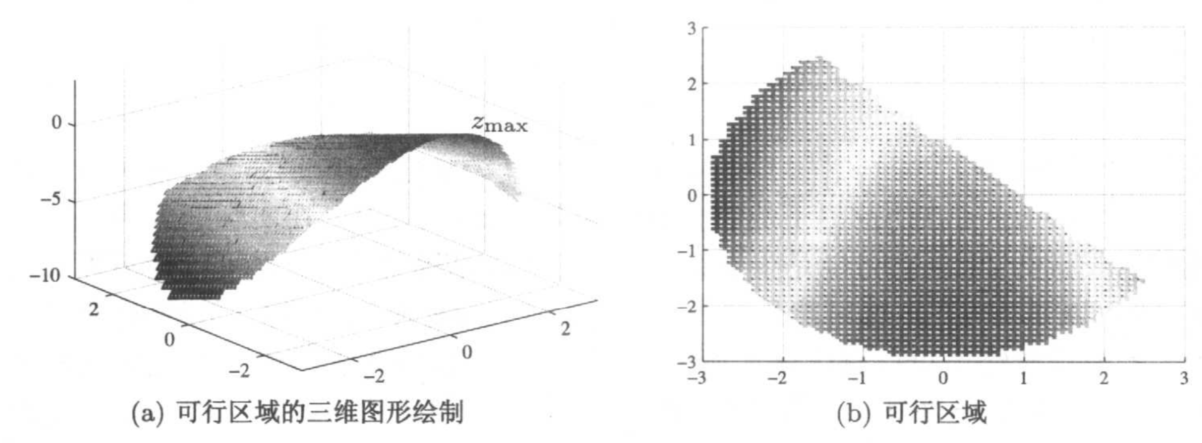
```
>> [x1,x2]=meshgrid(-3:.1:3);  % 生成网格型矩阵
   z=-x1.^2-x2;                 % 计算出矩阵上各点的高度
```

引入了约束条件，则在图形上需要将约束条件以外的点剔除掉。在 MATLAB 绘图技术中有一个技巧是找出这些点的横纵坐标值，将其函数值设置成不定式 NaN 即可。这样可以使用如下的语句进行求解。

```
>> i=find(x1.^2+x2.^2>9); z(i)=NaN; % 将 x1^2+x2^2>9 点设置成 NaN
   i=find(x1+x2>1); z(i)=NaN;   % 将 x1+x2>1 点的函数值设置为 NaN
   surf(x1,x2,z); shading interp;
```

该语句可以直接绘制出如图 5-15 (a) 所示的三维图形，若想从上向下观察该图形，则可以使用 view(0,90) 命令，这样可以得出如图 5-15 (b) 所示的二维投影图。图形上的区域为相应最优化问题的可行区域，即满足约束条件的区域。该区域内

对应目标函数的最大值就是原问题的解，故从图形可以直接得出结论，问题的解为 $x_1 = 0, x_2 = -3$，用 max(z(:)) 可以得出最大值为 3。



(a) 可行区域的三维图形绘制　　(b) 可行区域

图 5-15　二维最优化问题的图解法

对于一般的一元问题和二元问题，可以用图解法直接得出问题的最优解。但对于一般的多元问题和较复杂的问题，则不适合用图解法求解，而只能用数值解的方法进行求解，也没有检验全局最优性的方法。

5.3.2　线性规划问题的计算机求解

线性规划问题是一类特殊的问题，也是最简单的有约束最优化问题。在线性规划中，目标函数和约束函数都是线性的，其整个问题的数学描述为

$$\min_{\boldsymbol{x} \text{ s.t. } \begin{cases} \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B} \\ \boldsymbol{A}_{\mathrm{eq}}\boldsymbol{x} = \boldsymbol{B}_{\mathrm{eq}} \\ \boldsymbol{x}_{\mathrm{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\mathrm{M}} \end{cases}} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{x} \tag{5-3-2}$$

为描述原问题的方便及求解的高效性起见，这里的约束条件已经进一步细化为线性等式约束 $\boldsymbol{A}_{\mathrm{eq}}\boldsymbol{x} = \boldsymbol{B}_{\mathrm{eq}}$，线性不等式约束 $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B}$，$\boldsymbol{x}$ 变量的上界向量 $\boldsymbol{x}_{\mathrm{M}}$ 和下界向量 $\boldsymbol{x}_{\mathrm{m}}$，使得 $\boldsymbol{x}_{\mathrm{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\mathrm{M}}$。

对不等式约束来说，MATLAB 定义的标准型是"⩽"关系式。如果约束条件中某个式子是"⩾"关系式，则在不等号两边同时乘以 -1 就可以转换成"⩽"关系式了。

线性规划是一类最简单的有约束最优化问题，求解线性规划问题有多种算法。其中，单纯形法是最有效的一种方法，MATLAB 的最优化工具箱中实现了该算法，提供了求解线性规划问题的 linprog() 函数。该函数的调用格式为

$$[\boldsymbol{x}, f_{\mathrm{opt}}, \mathtt{flag}, \mathtt{c}] = \mathtt{linprog}(\boldsymbol{f}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\mathrm{eq}}, \boldsymbol{B}_{\mathrm{eq}}, \boldsymbol{x}_{\mathrm{m}}, \boldsymbol{x}_{\mathrm{M}}, \boldsymbol{x}_0, \mathtt{OPT}, p_1, p_2, \cdots)$$

其中，$\boldsymbol{f}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_{\mathrm{eq}}$, $\boldsymbol{B}_{\mathrm{eq}}$, $\boldsymbol{x}_{\mathrm{m}}$, $\boldsymbol{x}_{\mathrm{M}}$ 与前面约束与目标函数公式中的记号是完全一

致的，$\boldsymbol{x}_0$ 为初始搜索点。各个矩阵约束如果不存在，则应该用空矩阵来占位。OPT 为控制选项，该函数还允许使用附加参数 $p_1, p_2, \cdots$。最优化运算完成后，结果将在变量 $\boldsymbol{x}$ 中返回，最优化的目标函数将在 $f_{\rm opt}$ 变量中返回。我们将通过下面的例子来演示线性规划的求解问题。

例 5-26 试求解下面的线性规划问题。

$$\min_{\boldsymbol{x}\ {\rm s.t.}\left\{\begin{array}{l}2x_2+x_3+4x_4+2x_5\leqslant 54\\3x_1+4x_2+5x_3-x_4-x_5\leqslant 62\\x_1,x_2\geqslant 0,x_3\geqslant 3.32,x_4\geqslant 0.678,x_5\geqslant 2.57\end{array}\right.} (-2x_1-x_2-4x_3-3x_4-x_5)$$

求解 从给出的式子可以看出，其目标函数可以用系数向量 $\boldsymbol{f}=[-2,-1,-4,-3,-1]^{\rm T}$ 表示，不等式约束有两个，即

$$\boldsymbol{A}=\begin{bmatrix}0 & 2 & 1 & 4 & 2\\3 & 4 & 5 & -1 & -1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}54\\62\end{bmatrix}$$

另外，由于没有等式约束，故可以定义 $\boldsymbol{A}_{\rm eq}$ 和 $\boldsymbol{B}_{\rm eq}$ 为空矩阵。由给出的数学问题还可以看出，$\boldsymbol{x}$ 的下界可以定义为 $\boldsymbol{x}_{\rm m}=[0,0,3.32,0.678,2.57]^{\rm T}$，且对上界没有限制，故可以将其写成空矩阵。由前面的分析，可以给出如下的 MATLAB 命令来求解线性规划问题，并立即得出结果。

```
>> f=-[2 1 4 3 1]'; A=[0 2 1 4 2; 3 4 5 -1 -1];
   B=[54; 62]; Ae=[]; Be=[]; xm=[0,0,3.32,0.678,2.57];
   ff=optimset; ff.TolX=1e-15; ff.TolFun=1e-20; ff.TolCon=1e-20;
   [x,f_opt,key,c]=linprog(f,A,B,Ae,Be,xm,[],[],ff)
```

最优化问题的解为 $\boldsymbol{x}=[19.785,0,3.32,11.385,2.57]^{\rm T}$，最优值为 -89.575。从列出的结果看，由于 key 值为 1，故求解是成功的。以上只用了 5 步就得出了线性规划问题的解，可见求解程序功能是很强大的，可以很容易得出线性规划问题的解。

例 5-27 考虑下面的线性规划问题，试用 MATLAB 的最优化工具箱求解此问题。

$$\max_{\boldsymbol{x}\ {\rm s.t.}\left\{\begin{array}{l}x_1/4-60x_2-x_3/50+9x_4\leqslant 0\\-x_1/2+90x_2+x_5/50-3x_4\geqslant 0\\x_3\leqslant 1,x_1\geqslant -5,x_2\geqslant -5,x_3\geqslant -5,x_4\geqslant -5\end{array}\right.} \left[\frac{3}{4}x_1-150x_2+\frac{1}{50}x_3-6x_4\right]$$

求解 原问题中应该求解的是最大值问题，所以需要首先将之转换成最小化问题，即将原目标函数乘以 -1，则目标函数将改写成 $-3x_1/4+150x_2-x_3/50+6x_4$。套用线性规划的格式可以得出 $\boldsymbol{f}^{\rm T}$ 向量为 $[-3/4,150,-1/50,6]$。

再分析约束条件，可见，由最后一条可以写成 $x_i\geqslant -5$，所以可确定自变量的最小值向量为 $\boldsymbol{x}_{\rm m}$=[−5;−5;−5;−5]。类似地，还能写出自变量的最大值向量为

x_M=[Inf;Inf;1;Inf]，其中 Inf 表示 $+\infty$。约束条件的前两条均为不等式约束，其中第 2 条为 $\geqslant$ 表示，需要将两端均乘以 -1，转换成 $\leqslant$ 不等式，这样可以写出不等式约束为

$$A=\begin{bmatrix}1/4 & -60 & -1/50 & 9\\ 1/2 & -90 & -1/50 & 3\end{bmatrix},\ B=\begin{bmatrix}0\\0\end{bmatrix}$$

由于原问题中没有等式约束，故应该令 $A_{\rm eq}$=[]，$B_{\rm eq}$=[]。最终可以输入如下的命令来求解此最优化问题，得出原问题的最优解。

```
>> f=[-3/4,150,-1/50,6]; Aeq=[]; Beq=[];
   A=[1/4,-60,-1/50,9; 1/2,-90,-1/50,3]; B=[0;0];
   xm=[-5;-5;-5;-5]; xM=[Inf;Inf;1;Inf]; ff=optimset;
   ff.TolX=1e-15; ff.TolFun=1e-20; TolCon=1e-20; ff.Display='iter';
   [x,f_opt,key,c]=linprog(f,A,B,Aeq,Beq,xm,xM,[0;0;0;0],ff)
```

可见，经过 10 步迭代，就能以很高精度得出原问题的最优解为 $x=[-5,-0.1947,1,-5]^{\rm T}$，目标函数的最优值为 -55.47。

例 5-28 在一些研究领域中经常遇到下面类型的线性规划问题，自变量不是前面介绍的单下标的，而是双下标或多重下标的，试求解这样的线性规划问题。

$$\min\ 2800(x_{11}+x_{21}+x_{31}+x_{41})+4500(x_{12}+x_{22}+x_{32})+6000(x_{13}+x_{23})+7300x_{14}$$
$$x\ \text{s.t.}\begin{cases}x_{11}+x_{12}+x_{13}+x_{14}\geqslant 15\\ x_{12}+x_{13}+x_{14}+x_{21}+x_{22}+x_{23}\geqslant 10\\ x_{13}+x_{14}+x_{22}+x_{23}+x_{31}+x_{32}\geqslant 20\\ x_{14}+x_{23}+x_{32}+x_{41}\geqslant 12\\ x_{ij}\geqslant 0,\ (i=1,2,3,4,j=1,2,3,4)\end{cases}$$

求解 这样的问题显然不能直接用前面介绍的方法直接求解，而应该首先将原问题转换成单下标自变量的最优化问题。为做这样的转换，应该重新选定变量，例如令 $x_1=x_{11},x_2=x_{12},x_3=x_{13},x_4=x_{14},x_5=x_{21},x_6=x_{22},x_7=x_{23},x_8=x_{31},x_9=x_{32},x_{10}=x_{41}$，这样可以将原问题改写成

$$\min\ 2800(x_1+x_5+x_8+x_{10})+4500(x_2+x_6+x_9)+6000(x_3+x_7)+7300x_4$$
$$x\ \text{s.t.}\begin{cases}-(x_1+x_2+x_3+x_4)\leqslant -15\\ -(x_2+x_3+x_4+x_5+x_6+x_7)\leqslant -10\\ -(x_3+x_4+x_6+x_7+x_8+x_9)\leqslant -20\\ -(x_4+x_7+x_9+x_{10})\leqslant -12\\ x_i\geqslant 0,\ i=1,2,\cdots,12\end{cases}$$

这样就可以用下面的语句解出问题的最优解。

```
>> f=2800*[1 0 0 0 1 0 0 1 0 1]+4500*[0 1 0 0 0 1 0 0 1 0]+...
     6000*[0 0 1 0 0 0 1 0 0 0]+7300*[0 0 0 1 0 0 0 0 0 0];
```

```
A=-[1 1 1 1 0 0 0 0 0 0; 0 1 1 1 1 1 1 0 0 0;
    0 0 1 1 0 1 1 1 1 0; 0 0 0 1 0 0 1 0 1 1];
B=-[15; 10; 20; 12]; xm=[0 0 0 0 0 0 0 0 0 0]; Aeq=[]; Beq=[];
x=linprog(f,A,B,Aeq,Beq,xm)
```

得出 $x = [4.2069, 0, 0, 10.7931, 0, 0, 0, 8, 1.2069, 0.0000]$。将得出的结果再反代回双下标自变量，则可见 $x_{11} = 4.2069$，$x_{14} = 10.7931$，$x_{31} = 8$，$x_{32} = 1.2069$，其余的自变量均为 0。

5.3.3 二次型规划的求解

二次型规划问题是另一种简单的有约束最优化问题，其目标函数为 $\boldsymbol{x}$ 的二次型形式，约束条件仍然为线性不定式约束。一般二次型规划问题的数学表示为

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B} \\ \boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{B}_{\text{eq}} \\ \boldsymbol{x}_{\text{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\text{M}} \end{cases}} \left(\frac{1}{2}\boldsymbol{x}^{\text{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^{\text{T}}\boldsymbol{x}\right) \tag{5-3-3}$$

和线性规划问题相比，二次型规划目标函数中多了一个二次项 $\boldsymbol{x}^{\text{T}}\boldsymbol{H}\boldsymbol{x}$ 来描述 x_i^2 和 x_ix_j 项。MATLAB 的最优化工具箱提供了求解二次型规划问题的 `quadprog()` 函数，该函数的调用格式为

$[\boldsymbol{x}, f_{\text{opt}}, \texttt{flag}, \texttt{c}] = \texttt{quadprog}(\boldsymbol{H}, \boldsymbol{f}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\text{eq}}, \boldsymbol{B}_{\text{eq}}, \boldsymbol{x}_{\text{m}}, \boldsymbol{x}_{\text{M}}, \boldsymbol{x}_0, \texttt{OPT}, p_1, p_2, \cdots)$

其中，函数调用时，$\boldsymbol{H}$ 为二次型规划目标函数中的 $\boldsymbol{H}$ 矩阵，其余各个变量与线性规划函数调用的完全一致。

例 5-29 试求解下面的四元二次型规划问题。

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1 + x_2 + x_3 + x_4 \leqslant 5 \\ 3x_1 + 3x_2 + 2x_3 + x_4 \leqslant 10 \\ x_1, x_2, x_3, x_4 \geqslant 0 \end{cases}} \left[(x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (x_4 - 4)^2\right]$$

求解 首先应该将原始问题写成二次型规划的模式。展开目标函数得

$$\begin{aligned} f(x) &= x_1^2 - 2x_1 + 1 + x_2^2 - 4x_2 + 4 + x_3^2 - 6x_3 + 9 + x_4^2 - 8x_4 + 16 \\ &= x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2x_1 - 4x_2 - 6x_3 - 8x_4 + 30 \end{aligned}$$

因为目标函数中的常数对最优化结果没有影响，所以可以放心地略去。这样就可以将二次型规划标准型中的 $\boldsymbol{H}$ 矩阵和 $\boldsymbol{f}^{\text{T}}$ 向量写为

$$\boldsymbol{H} = \text{diag}([2, 2, 2, 2]), \quad \boldsymbol{f}^{\text{T}} = [-2, -4, -6, -8]$$

从而可以给出下列 MATLAB 命令来求解二次型最优化问题。

```
>> f=[-2,-4,-6,-8]; H=diag([2,2,2,2]);
   OPT=optimset; OPT.LargeScale='off'; % 不使用大规模问题算法
   A=[1,1,1,1; 3,3,2,1]; B=[5;10]; Aeq=[]; Beq=[]; xm=zeros(4,1);
   [x,f_opt]=quadprog(H,f,A,B,Aeq,Beq,xm,[],[],OPT)
```

得出的最优解为 $\boldsymbol{x}=[0,0.6667,1.6667.2.6667]^{\mathrm{T}}$，最优值为 -23.6667。

套用二次型规划标准型时，一定要注意 $\boldsymbol{H}$ 矩阵的生成，因为在式 (5-3-3) 中有一个 1/2 项，所以在本例中，$\boldsymbol{H}$ 矩阵对角元素是 2，而不是 1。另外，这里得出的目标函数实际上不是原始问题中的最优函数，因为人为地除去了常数项。将得出的结果再补上已经除去了的常数项，则可以求出原问题目标函数的值为 6.3333。

5.3.4　一般非线性规划问题的求解

有约束非线性最优化问题的一般描述为

$$\min_{\boldsymbol{x}\ \mathrm{s.t.}\ \boldsymbol{G}(\boldsymbol{x})\leqslant\boldsymbol{0}} f(\boldsymbol{x}) \tag{5-3-4}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$。为求解方便，约束条件还可以进一步细化为线性等式约束、线性不等式约束、$\boldsymbol{x}$ 变量的上下界向量，还允许一般非线性函数的等式和不等式约束，这时原规划问题可以改写成

$$\min_{\boldsymbol{x}\ \mathrm{s.t.}\left\{\begin{array}{l}\boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B}\\ \boldsymbol{A}_{\mathrm{eq}}\boldsymbol{x}=\boldsymbol{B}_{\mathrm{eq}}\\ \boldsymbol{x}_{\mathrm{m}}\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_{\mathrm{M}}\\ \boldsymbol{C}(\boldsymbol{x})\leqslant\boldsymbol{0}\\ \boldsymbol{C}_{\mathrm{eq}}(\boldsymbol{x})=\boldsymbol{0}\end{array}\right.} f(\boldsymbol{x}) \tag{5-3-5}$$

MATLAB 最优化工具箱中提供了一个 `fmincon()` 函数，专门用于求解各种约束下的最优化问题。该函数的调用格式为

$[\boldsymbol{x},f_{\mathrm{opt}},\mathtt{flag},\mathtt{c}]=\mathtt{fmincon}(\mathtt{F},\boldsymbol{x}_0,\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\mathrm{eq}},\boldsymbol{B}_{\mathrm{eq}},\boldsymbol{x}_{\mathrm{m}},\boldsymbol{x}_{\mathrm{M}},\mathtt{CF},\mathtt{OPT},p_1,p_2,\cdots)$

其中，F 为给目标函数写的 M 函数或匿名函数，$\boldsymbol{x}_0$ 为初始搜索点。各个矩阵约束如果不存在，则应该用空矩阵来占位。CF 为给非线性约束函数写的 M 函数，OPT 为控制选项。最优化运算完成后，结果将在变量 $\boldsymbol{x}$ 中返回，最优化的目标函数将在 f_{opt} 变量中返回。和其他优化函数一样，选项 OPT 有时是很重要的。输入变量 CF_OPT 为约束条件对应的文件名，由于该文件需要返回两个量，所以不能用 `inline()` 函数和匿名函数来描述，只能编写 M-函数表示约束条件。

例 5-30 试求解下面的有约束最优化问题。

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1^2+x_2^2+x_3^2-25=0 \\ 8x_1+14x_2+7x_3-56=0 \\ x_1,x_2,x_3 \geqslant 0 \end{cases}} \left[1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3\right]$$

求解 分析给出的最优化问题可以发现，约束条件中含有非线性不等式，故而不能使用二次型规划的方式求解，必须用非线性规划的方式来求解。根据给出的问题可以直接写出目标函数为

```
>> f=@(x)[1000-x(1)^2-2*x(2)^2-x(3)^2-x(1)*x(2)-x(1)*x(3)];
```

同时，给出的两个约束条件均为等式约束，所以应该写出非线性约束函数为

```
function [c,ceq]=optcon1(x)
ceq=[x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; 8*x(1)+14*x(2)+7*x(3)-56];
c=[];
```

非线性约束函数返回变量分为 c 和 ceq 两个量，其中，前者为不等式约束的数学描述，后者为非线性等式约束，如果某个约束不存在，则应该将其值赋为空矩阵。这样的约束函数处理比早期版本的工具箱中处理更方便、规范。

描述了给出的非线性等式约束后，则 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\rm eq}, \boldsymbol{B}_{\rm eq}$ 都将为空矩阵了。另外，应该给出搜索初值向量 $\boldsymbol{x}_{\rm m} = [0,0,0]^{\rm T}$，因此，可以调用 fmincon() 函数求解此约束最优化问题。

```
>> ff=optimset; ff.LargeScale='off'; ff.Display='iter';
   ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
   x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
   [x,f_opt,c,d]=fmincon(f,x0,A,B,Aeq,Beq,xm,xM,@optcon1,ff)
```

从而搜索出最优解为 $\boldsymbol{x} = [3.5121, 0.2170, 3.5522]^{\rm T}$，最优值为 961.7152。

考虑到第 2 个约束条件实际上是线性等式约束，所以能将其从非线性约束函数中除去，故可以将非线性约束函数进一步简化为

```
function [c,ceq]=optcon2(x)
ceq=x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; c=[];
```

线性等式约束可以由相应的矩阵定义出来，这时可以用下面的命令求解原始的最优化问题，且可以得出和前面完全一致的结果。

```
>> x0=[1;1;1]; Aeq=[8,14,7]; Beq=56;
   [x,f_opt,c,d]=fmincon(f,x0,A,B,Aeq,Beq,xm,xM,@optcon2,ff)
```

例 5-31 考虑例 5-30 中给出的最优化问题，试利用梯度求解最优化问题，并比较和原例子中所用方法的优劣。

求解 由给出的目标函数 $f(x)$ 可以立即求出下面的梯度函数 (或 Jocobi 矩阵)。

```
>> syms x1 x2 x3; f=1000-x1*x1-2*x2*x2-x3*x3-x1*x2-x1*x3;
   J=jacobian(f,[x1,x2,x3])
```

得出的 Jacobi 矩阵可以写成

$$J=\left[\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\frac{\partial f}{\partial x_3}\right]^{\mathrm{T}}=\begin{bmatrix}-2x_1-x_2-x_3\\-4x_2-x_1\\-2x_3-x_1\end{bmatrix}$$

有了梯度，则可以重新改写目标函数为

```
function [y,Gy]=fun2(x)
y=1000-x(1)*x(1)-2*x(2)*x(2)-x(3)*x(3)-x(1)*x(2)-x(1)*x(3);
Gy=-[2*x(1)+x(2)+x(3); 4*x(2)+x(1); 2*x(3)+x(1)];
```

其中，Gy 表示目标函数的梯度向量。再调用最优化求解函数将得出下面的结果:

```
>> x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
   ff=optimset; ff.GradObj='on'; ff.LargeScale='off';
   ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
   [x,f_opt,c,d]=fmincon(@fun2,x0,A,B,Aeq,Beq,xm,xM,@optcon1,ff);
```

得出最优化问题的解为 $x=[3.5121,0.2170,3.5522]^{\mathrm{T}}$，目标函数的最优值为 961.7152。可见，若已知目标函数的偏导数，则仅需 86 步目标函数的调用就求出原问题的解，比前面需要的步数 (113 步) 明显减少。但考虑求取和编写梯度函数所需的时间，实际需要的时间可能更多。注意，若已知梯度函数，则应该将 GradObj 选项设置成 'on'，否则不能识别该梯度。

5.3.5 有约束最优化问题在最优控制中的应用

如果能获得系统的仿真数据，则可以由这些数据定义出重要的实用约束条件，然后利用有约束最优化问题的求解，将满足约束条件的最优控制器设计出来。例如，可以指定系统的超调量作为指标，可以用 fmincon() 得出最优化问题的解。当然，也可以设置其他的约束条件。

例 5-32 考虑受控对象 $G(s)=\dfrac{(-0.2s+1)\left(1+\dfrac{\mathrm{e}^{-s}}{s+1}\right)}{(s+1)^2}$，并期望 $|u(t)|\leqslant 1$，超调量 $\sigma\leqslant 3\%$，试设计出最优 PID 控制器。

求解 这样的受控对象模型用普通 PID 控制器整定方法[10]是很难控制的。由给出的受控对象模型可见，该模型可以重新写成 $G(s)=\left(1+\dfrac{\mathrm{e}^{-s}}{s+1}\right)\dfrac{-0.2s+1}{(s+1)^2}$，其中括号内部的部分可以认为是两个子模型的并联，而括号内部的模型和后面的模型是串联连

接。这样可以构造出如图 5-16 所示的 PID 控制仿真模型。选择仿真终止时间 $t_f = 10$，则可以建立起本问题的目标函数。

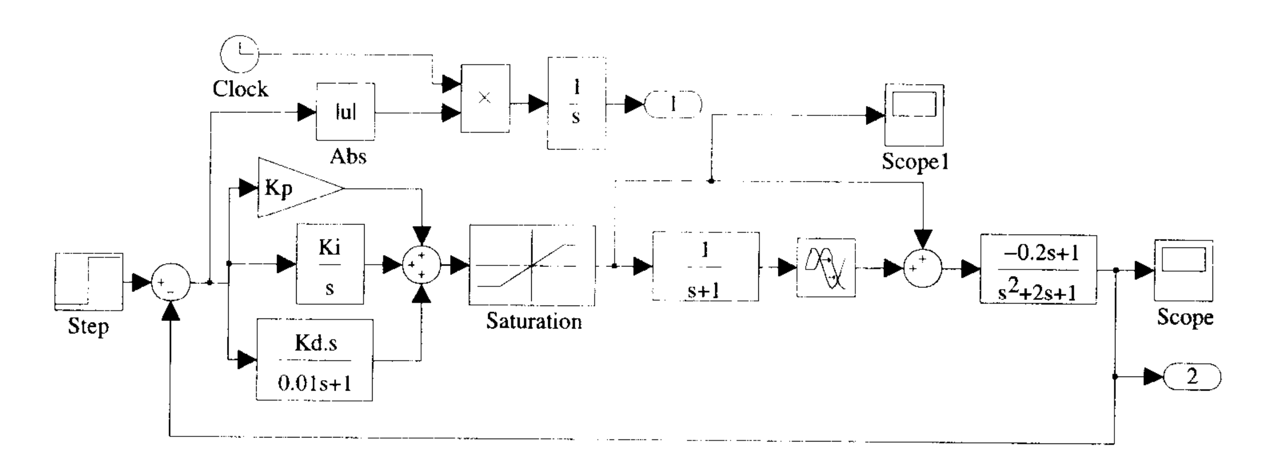


图 5-16 PID 控制的 Simulink 模型 (文件名：c5mopta.mdl)

```
function y=c5optfun2(x)
assignin('base','Kp',x(1)); assignin('base','Ki',x(2));
assignin('base','Kd',x(3));
[t,x1,y1]=sim('c5mopta.mdl',[0,25]); y=y1(end,1);
```

这样，由前面介绍的无约束最优化方法可以得出最优控制器，$G_1(s) = 1.1646 + \dfrac{0.2434}{s} - \dfrac{0.0641s}{0.01s+1}$，并可以得出如图 5-17 (a) 所示的阶跃响应曲线。由得出的控制效果看，这样的输出信号超调量过大，超过 20%。

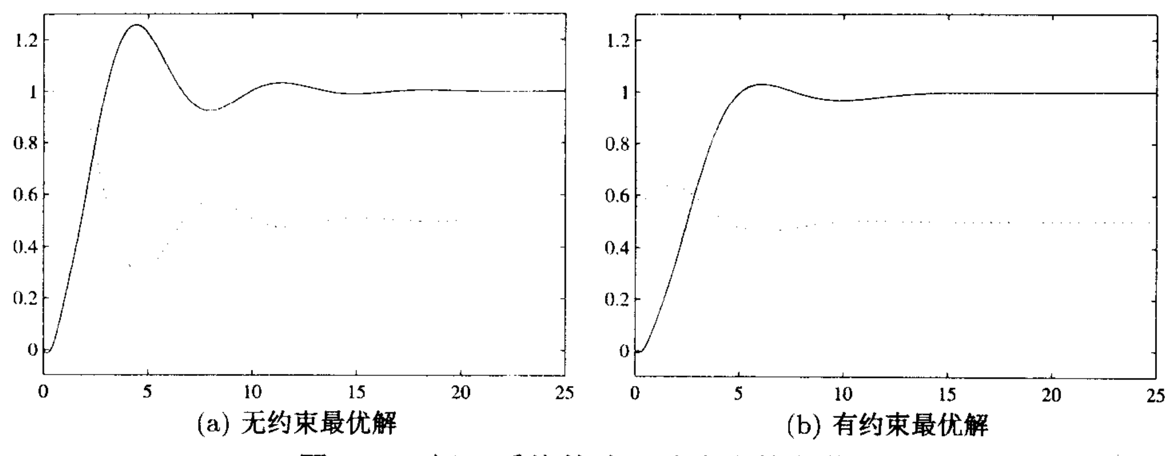
```
>> x0=[0.5 0.1 0]; x=fminunc(@c5optfun2,x0)
```

如果想减小超调量，使其满足要求的 $\sigma \leqslant 3\%$，可以引入如下的 MATLAB 函数描述约束

```
function [c,ce]=c5optcon2(x)
ce=[]; [t,x1,y1]=sim('c5mopta.mdl',[0,25]); c=max(y1(:,2))-1.03;
```

这样，由下面的语句可以设计出满足约束条件的 PID 控制器 $G_2(s) = 0.5175 + \dfrac{0.1910}{s} + \dfrac{0.1028}{0.01s+1}$。在新的控制器下，系统的闭环阶跃响应曲线如图 5-17 (b) 所示。可见，在新控制器的作用下，超调量满足要求，但系统响应速度稍有减慢。其实进一步观察由 PID 控制器给出的控制信号看，由新控制器产生的控制信号更好些。

```
>> x0=[0.5 0.1 0]; A=[]; B=[]; Ae=[]; Be=[]; xm=[0;0;0]; xM=[];
   x=fmincon(@c5optfun2,x0,A,B,Ae,Be,xm,xM,@c5optcon2)
```



(a) 无约束最优解　　(b) 有约束最优解

图 5-17　闭环系统的阶跃响应和控制信号

5.4　混合整数规划问题的计算机求解

在很多应用领域中，最优化问题的要求除了前面的满足约束条件的规则外，还需要使得全部和部分自变量取整数，这类问题又称为整数规划。其中，要求全部自变量均为整数的最优化问题又称为纯整数规划问题，部分自变量要求为整数的最优化问题又称为混合整数规划的问题。还有一类最优化问题，要求自变量只能是 0 或 1，这类规划问题又称为 0-1 规划问题。整数规划当然又分为线性整数规划问题和非线性整数规划问题。

5.4.1　整数线性规划问题的求解

整数线性规划的一般数学描述为

$$\min_{\boldsymbol{x}\ \text{s.t.}\ \begin{cases}\boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B}\\ \boldsymbol{A}_{\rm eq}\boldsymbol{x}=\boldsymbol{B}_{\rm eq}\\ \boldsymbol{x}_{\rm m}\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_{\rm M}\\ \hat{\boldsymbol{x}}\ \text{为整数}\end{cases}} \boldsymbol{f}^{\rm T}\boldsymbol{x} \tag{5-4-1}$$

其中 $\hat{\boldsymbol{x}}$ 为变量 $\boldsymbol{x}$ 的子集。如果已知自变量所在的区间，则理论上可以考虑用穷举方法列举出区间内所有的变量组合，逐个判定约束条件是否满足，从满足的组合中逐个求取函数的值并排序，由其最小值的对应关系可以简单地求解所需的自变量值。这个方法看似简单、直观，但对稍微多些自变量的情形是不可行的，因为这时计算量为天文数字。相应的数学问题又称为 NP 难 (non-polynomial hard) 问题，故穷举方法只适合于极有限的小规模问题。

MATLAB 自身没有提供整数线性规划的函数，但可以使用荷兰 Eindhoven 科技大学 Michel Berkelaar 等人开发的 LP_Solve 包中的 MATLAB 支持的 mex

文件。该程序据称可以求解多达 30000 个变量，50000 个约束条件的整数线性规划问题，该软件可以从网站 http://tech.groups.yahoo.com/group/lp_solve/ 下载，其调用格式为 [$\boldsymbol{x}$,how]=ipslv_mex($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{f}$,intlist,$\boldsymbol{x}_{\rm M}$,$\boldsymbol{x}_{\rm m}$,ctype)。其中，$\boldsymbol{A},\boldsymbol{B}$ 表示线性等式和不等式约束。和最优化工具箱中提供的函数不同，这里不要求用多个矩阵分别表示等式和不等式，而可以使用这两个矩阵表示等式、大于式和小于式，具体的约束式子由 ctype 变量控制，ctype(i) 的值为正、零与负分别对应 $\boldsymbol{a}_i\boldsymbol{x} \geqslant b_i$，$\boldsymbol{a}_i\boldsymbol{x} = b_i$ 和 $\boldsymbol{a}_i\boldsymbol{x} \leqslant b_i$。intlist 为整数约束标示，其第 i 个分量为 1 则表示需要 x_i 为整数。返回的 how 为得出解 $\boldsymbol{x}$ 的附加说明，其中，how= 0，表示结果为最优，为 2 表示无可行解，其余的值表示求解失败。

例 5-33 考虑例 5-26 中给出的线性规划问题，如果要求自变量 x_i 均为整数，则原来的问题就变成了整数线性规划问题，试求解该整数规划问题。

求解 依照 ipslv_mex() 函数的调用格式，建立起 $\boldsymbol{f}$ 向量，并建立起 $\boldsymbol{A},\boldsymbol{B}$ 矩阵和相应的 ctype()，并根据需要建立起 $\boldsymbol{x}_{\rm m}, \boldsymbol{x}_{\rm M}$ 向量，这样用下面的语句就可以求解出最优化问题。

```
>> f=-[2 1 4 3 1]'; A=[0 2 1 4 2; 3 4 5 -1 -1];
   intlist=ones(5,1); B=[54; 62]; ctype=[-1; -1];
   xm=[0,0,3.32,0.678,2.57]; xM=inf*ones(5,1);
   [x,b]=ipslv_mex(f,A,B,intlist,xM,xm,ctype)  % 返回 b=0 表示成功
```

最优解为 $\boldsymbol{x} = [19,0,4,10,5]^{\rm T}$。对于小规模问题，可以考虑采用穷举算法。人为假定 $\boldsymbol{x}_{\rm M}$ 的各个元素均为 20，采用下面语句就可以逐个求取函数值，得出所需的结果。

```
>> [x1,x2,x3,x4,x5]=ndgrid(1:20,0:20,4:20,1:20,3:20);
   i=find((2*x2+x3+4*x4+2*x5<=54) & (3*x1+4*x2+5*x3-x4-x5<=62));
   f=-2*x1(i)-x2(i)-4*x3(i)-3*x4(i)-x5(i); [fmin,ii]=sort(f);
   k=i(ii(1)); x=[x1(k),x2(k),x3(k),x4(k),x5(k)]
```

这样仍然能得出最优解为 $\boldsymbol{x} = [19,0,4,10,5]^{\rm T}$。然而这里有两个问题值得注意。其一，本算法得出的结果是 $x_1 \in [0,20]$ 区间的最小值，但这个概念不能随意拓展到此区间之外，如果想将 20 变为 30，在一般的计算机配置下都实现不了，因为所需内存过大，5 个变量的存储量为 $31^5 \times 5 \times 8/2^{20} = 1092.1\text{MB}$ 空间。所以在求解整数规划时不适合采用穷举算法。其二，除了得出的最优解之外，事实上还可以得出若干组合，使得该规划函数有次最优解。可以显示排序后的函数值如下：

```
>> fmin(1:10)'
   k=i(ii(1:15)); x=[x1(k),x2(k),x3(k),x4(k),x5(k),fmin(1:15)]
```

前十个最小的目标函数分别为 $[-89,-88,-88,-88,-88,-88,-88,-88,-87,-87]$。可见，函数的最小值为 -89。此外，还有若干个点的值为 -88，求出最优解的同时，还可以列出各个变量的次最优解，如表 5-3 所示。

表 5-3 最优解及部分次最优解

x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明
19	0	4	10	5	−89	最优	19	0	4	9	7	−88	次优	11	0	8	10	3	−87	次优
18	0	4	11	3	−88	次优	16	0	6	8	8	−88	次优	10	0	9	9	4	−87	次优
17	0	5	10	4	−88	次优	20	0	4	7	11	−88	次优	8	0	10	9	4	−87	次优
15	0	6	10	4	−88	次优	15	0	6	10	3	−87	次优	5	0	12	8	5	−87	次优
12	0	8	9	5	−88	次优	13	0	7	10	3	−87	次优	18	0	4	10	5	−87	次优

如果要求 x_1,x_4,x_5 为整数，其他两个变量为任意数，则原问题就变成了混合整数规划问题。用户应该修改一下 intlist 变量，将其设置为 intlist=[1,0,0,1,1]，则可以用下面的语句求出原问题的解。

```
>> intlist=[1,0,0,1,1];
   [x,b]=ipslv_mex(f,A,B,intlist,xM,xm,ctype)
```

这时混合整数规划问题的解为 $\boldsymbol{x}=[10,0,3.8,11,3]^{\mathrm{T}}$。

5.4.2 一般非线性整数规划问题与求解

前面的运算局限于线性规划问题的整数规划研究。在实际应用中经常需要求解非线性整数规划或混合规划问题，该领域中一种常用的算法是分枝定界 (branch and bound) 算法，具体算法在这里不详细介绍，只介绍一个基于该算法编写的现成函数 bnb20()，可以用来求解一般非线性整数规划的问题。该函数是荷兰 Groningen 大学的 Koert Kuipers 编写的，可以从 The MathWorks 网站上直接下载。该函数的调用格式为

$$[\mathrm{err},f,\boldsymbol{x}]=\mathrm{bnb20}(\mathrm{fun},\boldsymbol{x}_0,\mathrm{intlist},\boldsymbol{x}_{\mathrm{m}},\boldsymbol{x}_{\mathrm{M}},\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\mathrm{eq}},\boldsymbol{B}_{\mathrm{eq}},\mathrm{CFun})$$

其中，调用过程中的大部分输入变量与最优化工具箱函数几乎完全一致，该函数直接调用了最优化工具箱中的 fmincon() 函数，该函数还可以根据需要带附加参数，返回的变量 err 为函数的错误信息字符串，$\boldsymbol{x}$ 和 f 分别为最优解和其函数值。如果正确返回最优解，则 err 字符串为空字符串。该函数尚有不完全之处，因为给出的解往往不是很精确的整数，所以应该在该函数调用结束后给出下面的语句将其化成整数。

```
if length(err)==0, x(intlist==1)=round(x(intlist==1)), end
```

例 5-34 试用 bnb20() 函数求解例 5-33 中给出的线性整数规划问题。

求解 根据要求，可以编写出如下文件来描述目标函数

```
function f=c5miopt(x)
```

```
f=-[2 1 4 3 1]*x;
```

和前面介绍的线性规划问题求解不同，上限变量不能再选择为无穷大，而应该选择为较大的数值，例如均选择为 20000。同样，整数的下界也不能再选择为小数，而应该为整数，故也需要相应地变化，这样用下面的语句求解出的线性整数规划问题与例 5-33 得出的完全一致[1]。

```
>> A=[0 2 1 4 2; 3 4 5 -1 -1]; intlist=ones(5,1); Aeq=[]; Beq=[];
   B=[54; 62]; xm=[0,0,4,1,3]'; xM=20000*ones(5,1); x0=xm;
   [err,f,x]=bnb20('c5miopt',x0,intlist,xm,xM,A,B,Aeq,Beq);
   if length(err)==0, x(intlist==1)=round(x(intlist==1)), end
```

仍要求 x_1, x_4, x_5 为整数，其他两个变量为任意值，则应该修改一下 intlist 变量，将其设置为 intlist=[1,0,0,1,1]，则可以用下面的语句求出原问题的解

```
>> intlist=[1,0,0,1,1]'; xm(intlist==1)=round(xm(intlist==1));
   [errmsg,f,x]=bnb20('c5miopt',x0,intlist,xm,xM,A,B,Aeq,Beq);
   if length(err)==0, x(intlist==1)=round(x(intlist==1)), end
```

最优解为 $\boldsymbol{x} = [19, 0, 3.8, 11, 3]^{\mathrm{T}}$，仍与例 5-33 完全一致。

例 5-35 对著名的 Rosenbrock 函数稍加修改，试求解整数 x_1 和 x_2，使得

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} -100 \leqslant x_1 \leqslant 100 \\ -100 \leqslant x_2 \leqslant 100 \end{cases}} 100(x_2 - x_1^2)^2 - (4.5543 - x_2)^2$$

求解 由给定的目标函数可以编写出如下的 MATLAB 函数:

```
function y=c5mfun1(x)
y=100*(x(2)-x(1)^2)^2+(4.5543-x(1))^2;
```

这样由下面的语句可以得出问题的解

```
>> x0=[1;1]; xm=-100*[1;1]; xM=100*[1;1];
   A=[]; B=[]; Aeq=[]; Beq=[]; intlist=[1,1]';
   [errmsg,f,x]=bnb20('c5mfun1',x0,intlist,xm,xM,A,B,Aeq,Beq);
   if length(errmsg)==0, x=round(x), end; x=x'
```

由得出的解可以看出，开始时 x_2 稍有误差，经取整得 $x_1 = 5, x_2 = 25$。注意，在这样的问题求解中，搜索区域的选择也是很重要的，如果用户为节省时间选择了一个更小的区间，如 $x_{1,2} \in [-20, 20]$，则将得出如下结果，可见该值不是原问题的最优解。

```
>> xm=-20*[1;1]; xM=20*[1;1];
```

[1] 注意，因为兼容性原因，这里的目标函数不支持匿名函数和 inline() 函数描述，引用 M 函数也只能用单引号形式，不能用 @ 号。

```
[errmsg,f,x]=bnb20('c5mfun1',x0,intlist,xm,xM,A,B,Aeq,Beq);
if length(errmsg)==0, x=round(x), end; x=x'
```

这时得到的在小区域的最优解为 $x = [4, 16]^T$。

其实，对这样小规模的问题，选择较大的搜索范围，如 $x_{1,2} \in (-200, 200)$，用穷举搜索算法能立即得出问题的解，和上述结果一致。

```
>> [x1,x2]=meshgrid(-200:200); f=100*(x2-x1.^2).^2+(4.5543-x1).^2;
   [fmin,i]=sort(f(:));  x=[x1(i(1)),x2(i(1))]
```

5.4.3 0-1 规划问题求解

所谓 0-1 规划，即指自变量 x_i 的值或者为 0，或者为 1。所以求解 0-1 规划看起来很简单，让每个自变量 x_i 遍取 0,1，在得出的组合中选择既满足约束条件又使目标函数取最小值的项。而事实上，随着问题规模的增大，这样的计算量将按指数增长。例如，自变量的个数为 n，则需要执行的循环次数将为 2^n，在 n 较大时其值可能是个天文数字，故仍然需要考虑其他算法进行求解。

MATLAB 7.0 版本提供了一个新函数 bintprog()，可以用来求解 0-1 线性规划问题，但不能直接求解非线性 0-1 规划问题的求解。该函数的调用格式为

$$\boldsymbol{x}=\texttt{bintprog}(\boldsymbol{f},\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\rm eq},\boldsymbol{B}_{\rm eq})$$

该函数可以直接用来求解下面例子演示的 0-1 线性规划问题。

例 5-36 试求解下面给出的 0-1 线性规划问题。

$$\min_{\boldsymbol{x} \text{ s.t. } \begin{cases} x_1+2x_2-x_3\leqslant 2 \\ x_1+4x_2+x_3\leqslant 4 \\ x_1+x_2\leqslant 3 \\ 4x_2+x_3\leqslant 6 \end{cases}} (-3x_1+2x_2+5x_3)$$

求解 套用所需的最优化模型，可以立即求出 $\boldsymbol{f}$、$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵，这样可以给出如下的语句求解 0-1 线性规划问题，得出

```
>> f=[-3,2,-5]; A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6];
   x=bintprog(f,A,B,[],[])'
```

这样可以得出 0-1 规划的结果为 $x_1 = 1, x_2 = 0, x_3 = 1$。

对于小规模问题，当然可以采用下面语句，逐个判定约束条件并找出目标函数的值，通过排序即能得出所需的结果，且可以保证此结果为全局最优解。

```
>> [x1,x2,x3]=meshgrid([0,1]);
   i=find((x1+2*x2-x3<=2)&(x1+4*x2+x3<=4)&(x1+x2<=3)&(4*x1+x3<=6));
   f=-3*x1(i)+2*x2(i)-5*x3(i); [fmin,k]=sort(f);
   index=i(ii(1)); x=[x1(index),x2(index),x3(index)]
```

```
x=[x1(i(k)),x2(i(k)),x3(i(k))]; [x fmin] % 可以列出所有可行解
```

可以得出最优解 $x = [1, 0, 1]^{\mathrm{T}}$，对应的目标函数为 −8。除了最优解外还可以列举出其他所有的可行解 $[0, 0, 1]^{\mathrm{T}} \Rightarrow -5$, $[1, 0, 0]^{\mathrm{T}} \Rightarrow -3$, $[0, 0, 0]^{\mathrm{T}} \Rightarrow 0$, $[0, 1, 0]^{\mathrm{T}} \Rightarrow 2$。

调用前面的 ipslv_mex() 或 bnb20() 函数，设定上限 x_{m} 和下限 x_{M} 分别为零向量和幺向量，再求整数规划就能得出原问题的解。

例 5-37 试用 ipslv_mex() 函数求解例 5-36 给出的 0-1 线性规划问题。

求解 按照给出的问题，设定 x_{m} 和 x_{M} 分别为零向量和幺向量，这样可以给出如下的语句求解 0-1 整数规划问题，得出

```
>> f=[-3,2,-5]; xm=[0;0;0]; xM=[1;1;1]; intlist=[1;1;1];
   A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6]; C=-1*ones(4,1);
   [res,b]=ipslv_mex(f,A,B,intlist,xM,xm,C)
```

结果和前面得出的完全一致。事实上，分析给定的约束条件，可以发现后两个约束条件是冗余的，可以取消。

由于 MATLAB 及其最优化工具箱对整数规划的支持功能很有限，免费的 MATLAB 函数求解非线性混合整数规划的功能也不是很强大，故 TOMLAB Optimization 公司推出了基于 MATLAB 的第三方软件 TOMLAB❶，可以更好地求解混合整数规划问题。该软件是商品软件，使用起来也没有 MATLAB 最优化工具箱和前面介绍的免费工具那么简单，但其功能齐全，能够求解各类最优化问题，是当前基于 MATLAB 的最好的最优化工具。限于篇幅，本书不介绍该工具箱。

5.5 最优化问题求解在控制中的其他应用

最优化技术在控制系统研究中的应用还是比较广泛的，早期研究由于没有合适的计算机工具，并未得到太多的重视，随着 MATLAB 这类语言的日益普及，最优化技术在控制界的应用将越来越受到关注。除了前面介绍的最优控制器除数寻优之外，这里将介绍方程求解和最优化技术在线性系统模型最优降阶、非线性系统工作点计算以及线性化等方面的应用，还将介绍两个实用的最优控制程序及其使用方法。

5.5.1 线性系统的最优降阶研究

在实际的线性系统研究中，经常会遇到高阶的系统模型。在控制理论发展初期，由于没有强大的计算机工具，人们没有办法对高阶系统直接分析，为了能更好地找出系统的特性，通常可以采用模型降阶方法得出近似的低价模型。当然，随

❶ 该工具箱的下载网址为 http://www.tomlab.com，读者可以下载该工具箱的 21 天试用版。

着高水平计算机语言、软件的发展和普及，模型降阶这方面的要求就不那么显著了。另一方面，在控制器设计领域存在大量的基于模型低阶模型的控制器设计方法，如 Ziegler-Nichols 的 PID 控制器整定公式只适合于低阶系统的设计，所以有必要用低阶模型去逼近高阶受控对象，进行控制器设计是很重要的。

控制系统的模型降阶问题是首先在 1966 年由 Edward J. Davison 提出的[11]，经过几十年的发展，出现了各种各样的降阶算法及应用领域。这里只研究基于最优化技术的模型降阶方法。

1. 降阶模型的指标选择

对降阶效果可能有各种各样的定义和指标，但最直观的是按图 5-18 中给出的形式定义出降阶误差信号 $e(t)$，根据该误差信号，可以定义出一些指标，如 $J_{\rm ISE}=\displaystyle\int_0^\infty e^2(t){\rm d}t$，将其定义为目标函数，对其最小化，得出最优降阶模型。

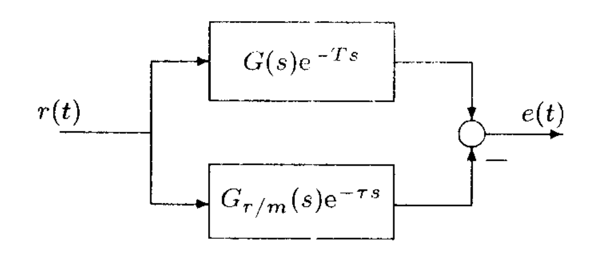


图 5-18 模型降阶误差信号

假设带有时间延迟的原始模型为

$$G(s){\rm e}^{-Ts}=\frac{b_1s^{n-1}+\cdots+b_{n-1}s+b_n}{s^n+a_1s^{n-1}+\cdots+a_{n-1}s+a_n}{\rm e}^{-Ts} \tag{5-5-1}$$

则降阶模型可以写成

$$G_{r/k}(s){\rm e}^{-\tau s}=\frac{\beta_1s^{r}+\cdots+\beta_rs+\beta_{r+1}}{s^k+\alpha_1s^{k-1}+\cdots+\alpha_{k-1}s+\alpha_k}{\rm e}^{-\tau s} \tag{5-5-2}$$

降阶误差信号的 Laplace 变换表达式为

$$E(s)=\left[G(s){\rm e}^{-Ts}-G_{r/m}(s){\rm e}^{-\tau s}\right]R(s) \tag{5-5-3}$$

其中 $R(s)$ 为输入信号 $r(t)$ 的 Laplace 变换式。

2. 次最优模型降阶算法[12]

利用最优化算法进行模型降阶的思路是很直观的。由前面定义的误差信号 $e(t)$ 和前面定义的 $J_{\rm ISE}$ 目标函数，通过参数最优化的方式寻优，找出降阶模型。对目标函数还可以进一步处理，例如对误差信号进行加权，引入新的误差信号

$h(t)=w(t)e(t)$，则可以定义出新的 ISE 指标：

$$\sigma_{\rm h}^2=\int_0^\infty h^2(t){\rm d}t=\int_0^\infty w^2(t)e^2(t){\rm d}t \tag{5-5-4}$$

若 $H(s)$ 为稳定的有理函数，则目标函数的值可以由 Åström 算法递推或 Lyapunov 方程求解。如果降阶模型或原始模型中含有时间延迟项，则用 Åström 算法不能直接求解，需要对延迟向采用 Padé 近似。因为对延迟系统采用近似的最优化来求解，所以这里称之为次最优降阶算法[12]。如果不含有延迟项，则称为最优降阶算法。

定义待定参数向量 $\boldsymbol{\theta}=(\alpha_1,\cdots,\alpha_{\rm m},\beta_1,\cdots,\beta_{r+1},\tau)$，则对一类给定输入信号可以定义出降阶模型的误差信号 $\widehat{e}(t,\boldsymbol{\theta})$，其中误差信号被显式地写成 $\boldsymbol{\theta}$ 的函数，这样可以定义出一个次最优降阶的目标函数为

$$J=\min_{\boldsymbol{\theta}}\left[\int_0^\infty w^2(t)\widehat{e}^2(t,\boldsymbol{\theta}){\rm d}t\right] \tag{5-5-5}$$

作者编写了 MATLAB 函数 opt_app()，可以用于求解带有时间延迟的次最优降阶模型，该函数的内容为

```
function G_r=opt_app(G_Sys,r,k,key,G0)
GS=tf(G_Sys); num=GS.num{1}; den=GS.den{1};
Td=totaldelay(GS); GS.ioDelay=0; GS.InputDelay=0;GS.OutputDelay=0;
if nargin<5,
   n0=[1,1]; for i=1:k-2, n0=conv(n0,[1,1]); end
   G0=tf(n0,conv([1,1],n0));
end
beta=G0.num{1}(k+1-r:k+1); alph=G0.den{1}; Tau=1.5*Td;
x=[beta(1:r),alph(2:k+1)]; if abs(Tau)<1e-5, Tau=0.5; end
dc=dcgain(GS); if key==1, x=[x,Tau]; end
y=opt_fun(x,GS,key,r,k,dc);
x=fminsearch('opt_fun',x,[],GS,key,r,k,dc);
alph=[1,x(r+1:r+k)]; beta=x(1:r+1); if key==0, Td=0; end
beta(r+1)=alph(end)*dc; if key==1, Tau=x(end)+Td; else, Tau=0; end
G_r=tf(beta,alph,'ioDelay',Tau);
```

该函数的调用格式为 $G_{\rm r}$=opt_app(G,r,k,key,G_0)，其中 G 和 $G_{\rm r}$ 分别为原始模型和降阶模型，r, k 为降阶模型的分子分母阶次，key 表明在降阶模型中是否需要延迟项，G_0 为最优化初值，可以忽略。该函数中调用的 opt_fun() 函数用于描述目标函数，其清单为

```
function y=opt_fun(x,G,key,r,k,dc)
ff0=1e10; a=[1,x(r+1:r+k)]; b=x(1:r+1); b(end)=a(end)*dc; g=tf(b,a);
if key==1, tau=x(end);
```

```
      if tau<=0, tau=eps; end, [n,d]=pade(tau,3); gP=tf(n,d);
   else, gP=1; end
   G_e=G-g*gP; G_e.num{1}=[0,G_e.num{1}(1:end-1)];
   [y,ierr]=geth2(G_e); if ierr==1, y=10*ff0; else, ff0=y; end
   function [v,ierr]=geth2(G)
   G=tf(G); num=G.num{1}; den=G.den{1}; ierr=0; v=0; n=length(den);
   if abs(num(1))>eps
      disp('System not strictly proper'); ierr=1; return
   else, a1=den; b1=num(2:length(num)); end
   for k=1:n-1
     if (a1(k+1)<=eps), ierr=1; return
     else,
        aa=a1(k)/a1(k+1); bb=b1(k)/a1(k+1); v=v+bb*bb/aa; k1=k+2;
        for i=k1:2:n-1, a1(i)=a1(i)-aa*a1(i+1); b1(i)=b1(i)-bb*a1(i+1);
   end, end, end
   v=sqrt(0.5*v);
```

例 5-38 已知原始系统的传递函数模型[13]

$$G(s)=\frac{1+8.8818s+29.9339s^2+67.087s^3+80.3787s^4+68.6131s^5}{1+7.6194s+21.7611s^2+28.4472s^3+16.5609s^4+3.5338s^5+0.0462s^6}$$

试得出其最优的三阶降阶模型。

求解 用下面的语句可以得出该模型的最优降阶模型

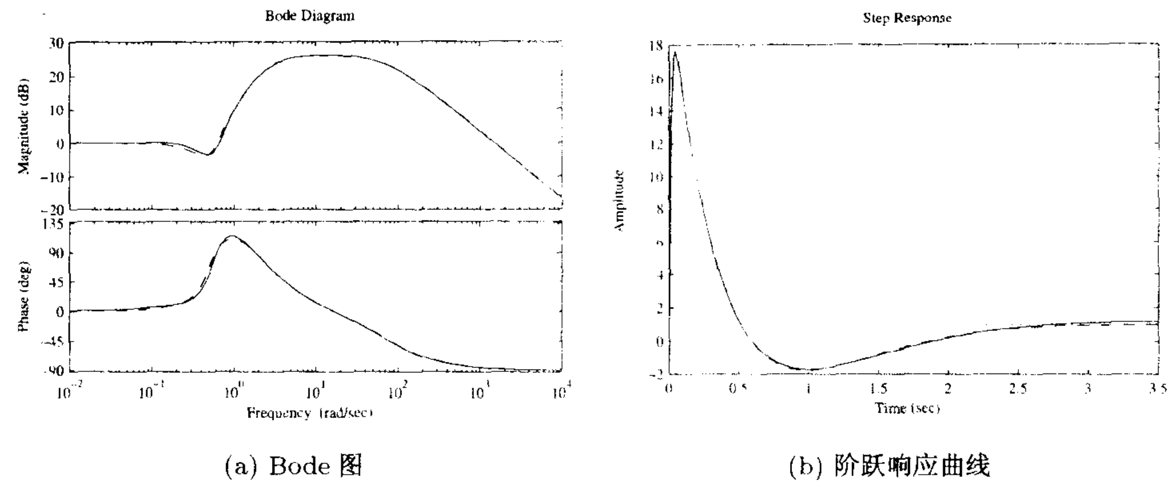
```
>> num=[68.6131,80.3787,67.087,29.9339,8.8818,1];
   den=[0.0462,3.5338,16.5609,28.4472,21.7611,7.6194,1];
   G=tf(num,den); Gr=zpk(opt_app(G,2,3,0))
   bode(G,Gr,'--'); figure; step(G,Gr,'--')
```

得出的最优降阶模型为 $G_{\mathrm{r}}(s)=\dfrac{1523.6536(s^2+0.3492s+0.2482)}{(s+74.85)(s^2+3.871s+5.052)}$。原系统与降阶模型的 Bode 图与阶跃响应曲线分别如图 5-19 (a)、(b) 所示，可见，由低阶模型 G_{r} 可以很好地逼近原高阶模型。

例 5-39 考虑系统模型[14] $G(s)=\dfrac{432}{(5s+1)(2s+1)(0.7s+1)(s+1)(0.4s+1)}$，试得出合适的降阶模型。

求解 由于原系统相对阶为 5，分母的阶次远高于分子的阶次，所以应该采用带有时间延迟的模型结构去逼近可能得出比较合理的近似。由下面的 MATLAB 语句则可以得出带延迟和不带延迟的次最优降阶模型

```
>> s=zpk('s'); G=432/((5*s+1)*(2*s+1)*(0.7*s+1)*(s+1)*(0.4*s+1));
   Grd=zpk(opt_app(G,0,2,1)), Gr=zpk(opt_app(G,1,2,0))
   step(G,Gr,'--',Grd,':')
```



(a) Bode 图　　(b) 阶跃响应曲线

图 5-19　原系统与降阶模型的比较

两个不同的最优降阶模型分别为

$$G_{\rm rd}(s) = \frac{31.4907}{(s+0.3283)(s+0.222)}{\rm e}^{-1.5s}, \quad G_{\rm r}(s) = \frac{-3.3801(s-5.229)}{s^2+0.353s+0.04092}$$

图 5-20 中比较了两个降阶模型和原系统在阶跃响应上的差异，可见，带有延迟的次最优降阶模型能很好地逼近原高阶系统的特性。

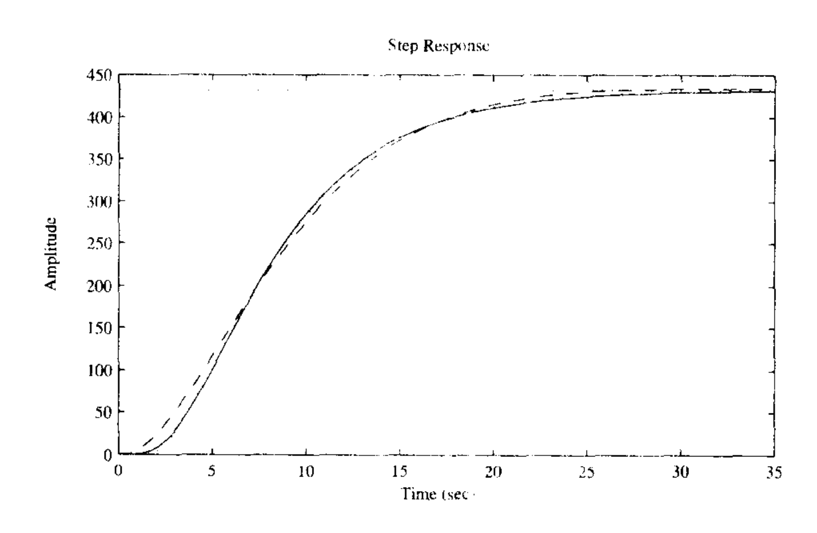


图 5-20　不同降阶模型的阶跃响应比较

5.5.2 非线性系统的线性化

比起非线性系统来说，线性系统更易于分析与设计．然而在实际应用中经常存在非线性系统，严格说来，所有的系统都含有不同程度的非线性成分。在这样的情况下，经常需要对非线性系统进行某种线性近似，从而简化系统的分析与设计。系统的线性化是提取线性系统特征的一种有效方法。系统的线性化实际上是

在系统的工作点附近的邻域内提取系统的线性特征，从而对系统进行分析设计的一种方法。

考虑下面给出的非线性系统的一般格式

$$\dot{x}_i(t) = f_i(x_1, x_2, \cdots, x_n, \boldsymbol{u}, t),\ \ i = 1, 2, \cdots, n \tag{5-5-6}$$

所谓系统的工作点，就是当系统状态变量导数趋于 0 时的状态变量的值。系统的工作点可以通过求取式 (5-5-6) 中非线性方程的方法得出：

$$f_i(x_1, x_2, \cdots, x_n, \boldsymbol{u}, t) = 0,\ \ i = 1, 2, \cdots, n \tag{5-5-7}$$

该方程可以采用数值算法求解，MATLAB 中提供了 Simulink 模型的工作点求取的实用函数 `trim()`，其调用格式为 `[`$\boldsymbol{x}$`,`$\boldsymbol{u}$`,`$\boldsymbol{y}$`,`$\boldsymbol{x}_\mathrm{d}$`]=trim(模型名,`$\boldsymbol{x}_0$`,`$\boldsymbol{u}_0$`)`，其中“模型名”为 Simulink 模型的文件名，变量 $\boldsymbol{x}_0$, $\boldsymbol{u}_0$ 为数值算法所要求的起始搜索点，是用户应该指定的状态初值和工作点的输入信号。对不含有非线性环节的系统来说，则不需要初始值 $\boldsymbol{x}_0$, $\boldsymbol{u}_0$ 的设定。调用函数之后，实际的工作点在 $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{y}$ 变量中返回，而状态变量的导数值在变量 $\boldsymbol{x}_\mathrm{d}$ 中返回。从理论上讲，状态变量在工作点处的一阶导数都应该等于 0。

得到工作点 $\boldsymbol{x}_0$ 后，非线性系统在此工作点附近，在 $\boldsymbol{u}_0$ 输入信号作用下可以近似地表示成

$$\Delta\dot{x}_i = \sum_{j=1}^{n} \left.\frac{\partial f_i(\boldsymbol{x}, \boldsymbol{u})}{\partial x_j}\right|_{\boldsymbol{x}_0, \boldsymbol{u}_0} \Delta x_j + \sum_{j=1}^{p} \left.\frac{\partial f_i(\boldsymbol{x}, \boldsymbol{u})}{\partial u_j}\right|_{\boldsymbol{x}_0, \boldsymbol{u}_0} \Delta u_j \tag{5-5-8}$$

选择新的状态变量，令 $\boldsymbol{z}(t) = \Delta\boldsymbol{x}(t)$，且 $\boldsymbol{v}(t) = \Delta\boldsymbol{u}(t)$，则得出线性模型

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{A}_\mathrm{l}\boldsymbol{z}(t) + \boldsymbol{B}_\mathrm{l}\boldsymbol{v}(t) \tag{5-5-9}$$

该模型称为线性化模型，其中

$$\boldsymbol{A}_\mathrm{l} = \begin{bmatrix} \partial f_1/\partial x_1 & \cdots & \partial f_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \cdots & \partial f_n/\partial x_n \end{bmatrix},\ \ \boldsymbol{B}_\mathrm{l} = \begin{bmatrix} \partial f_1/\partial r_1 & \cdots & \partial f_1/\partial r_p \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial r_1 & \cdots & \partial f_n/\partial r_p \end{bmatrix} \tag{5-5-10}$$

MATLAB 中还给出了 Simulink 模型线性化的 `linmod2()` 等函数，用以在工作点附近提取系统的线性化模型，这些函数可以直接获得系统的状态方程模型，其调用格式归纳如下：

```
[A,B,C,D]=linmod2(模型名,x0,u0)    % 一般连续系统线性化
[A,B,C,D]=linmod(模型名,x0,u0)     % 一般连续延迟系统线性化
[A,B,C,D]=dlinmod(模型名,x0,u0)    % 含离散环节的系统线性化
```

其中 $\boldsymbol{x}_0$, $\boldsymbol{u}_0$ 为工作点的状态与输入值，可以由 `trim()` 函数求出。对只由线性模块构成的 Simulink 模型来说，可以省略这两个参数，调用了本函数后，将自动返

回从输入端子到输出端子间的线性状态方程模型。`linmod()` 和 `linmod2()` 二者功能相似，但算法不同，前者可以处理延迟环节的 Padé 近似，而后者不能。

例 5-40 考虑例 4-37 中给出的多变量系统模型，试提取其状态方程模型。

求解 提取一个线性 Simulink 模型的状态方程就是对其进行线性化。如果想对该模型进行线性化，则需要将原系统 Simulink 框图中的阶跃输入用输入端子取代，更简单地，原系统中使用了阶跃模块和 Mux 模块，在线性化时将其统一化简成一个输入端子即可，因为输入端子模块支持向量型信号。另外，为使得含有纯时间延迟的系统能正确近似，还应该设置一下延迟模块的 Padé 近似阶次。双击时间延迟模块，将 Padé order (for linearization) 栏目填写上 2，就可以自动用二阶 Padé 近似取代原来的时间延迟环节了。最终得出的改写后多变量系统框图如图 5-21 所示。

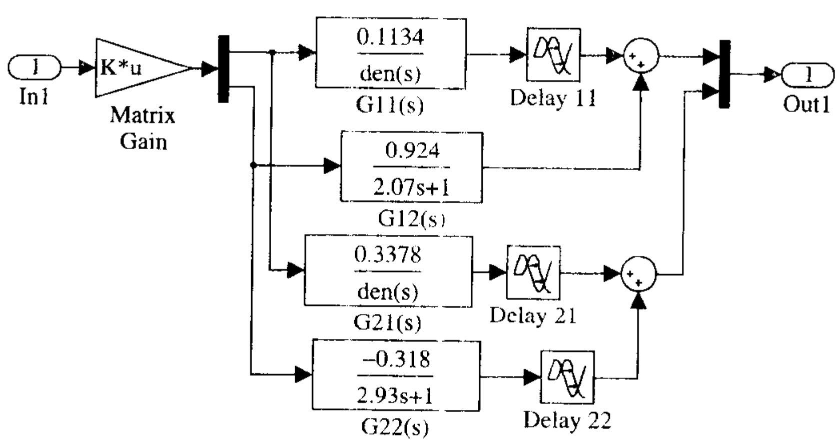


图 5-21 改写后的多变量系统 Simulink 模型 (文件名：c5mmdly1.mdl)

定义了 Simulink 框图，则需要用下面的语句进行系统的线性化，得出线性状态方程模型。由线性化模型得出的阶跃响应曲线如图 5-22 所示，其结果与精确的仿真模型得出的结果很接近。

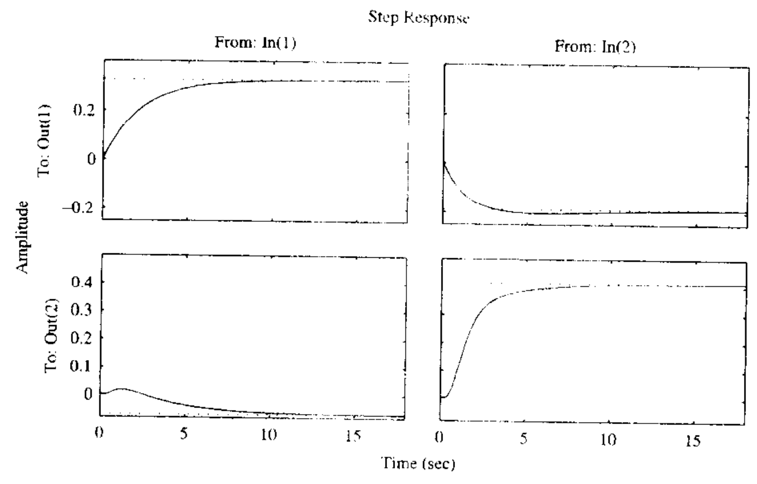


图 5-22 近似仿真与精确仿真阶跃曲线的比较

```
>> Kp=[0.1134,0.924; 0.3378,-0.318];
   [A,B,C,D]=linmod('c5mmdly1'), % 注意：延迟系统不能采用 linmod2()
   step(ss(A,B,C,D))
```

可以得出线性化模型的状态方程矩阵为

$$A=\begin{bmatrix} -8.3333 & -23.148 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0637 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & -0.4831 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & -20 & -133.33 & 0 & 0 & 0 & 0 & 0 & 0.9357 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & -4.6512 & -7.2111 & 0 & 0 & 0 & 0 & -0.10853\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.5169 & -0.5618 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3.0194 & -2.7701 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.3413 \end{bmatrix}$$

$$B^{\mathrm{T}}=\begin{bmatrix} 0 & 0 & 0.3378 & 0 & 0 & 0 & 0 & 0.1134 & 0 & 0.1134 & 0 & 0.3378\\ 0 & 0 & -0.318 & 0 & 0 & 0 & 0 & 0.924 & 0 & 0.924 & 0 & -0.318 \end{bmatrix}$$

$$C=\begin{bmatrix} -16.667 & 0 & 0.44638 & 0 & 0 & 0 & 0 & 0 & 0.063708 & 0 & 0 & 0\\ 0 & 0 & 0 & -40 & 0 & -9.3023 & 0 & 0 & 0 & 0 & 0.93573 & -0.10853 \end{bmatrix}$$

例 5-41 考虑如图 5-23 所示经典的计算机控制系统模型[15]，其中，控制器模型是离散模型，采样周期为 T 秒，ZOH 为零阶保持器，而受控对象模型为连续模型，假设受控对象和控制器都已经给定

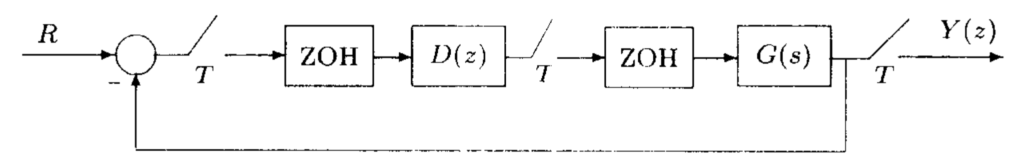


图 5-23　计算机控制系统框图

$$G(s)=\frac{a}{s(s+1)},\quad D(z)=\frac{1-\mathrm{e}^{-T}}{1-\mathrm{e}^{-0.1T}}\frac{z-\mathrm{e}^{-0.1T}}{z-\mathrm{e}^{-T}}$$

其中 $a=0.1$，试求其等效的传递函数模型。

求解　对这样的系统来说，直接写成微分方程形式再进行仿真的方法是不可行的，因为其中既有连续环节，又有离散环节，不可能直接写出系统的微分方程模型。若对该系统进行线性化，需要在其 Simulink 仿真模型中用输入端子和输出端子表示系统的输入和输出端口，构造出如图 5-24 所示的 Simulink 模型。

这样系统的等效连续传递函数模型可以由下面的语句进行线性化

```
>> [A,B,C,D]=dlinmod('c5mcomp2'); zpk(ss(A,B,C,D,'Ts',0.2))
```

得出的线性化模型为 $G(z)=\dfrac{0.018187(z+0.9934)(z-0.9802)}{(z-0.9802)(z^2-1.801z+0.8368)}$。

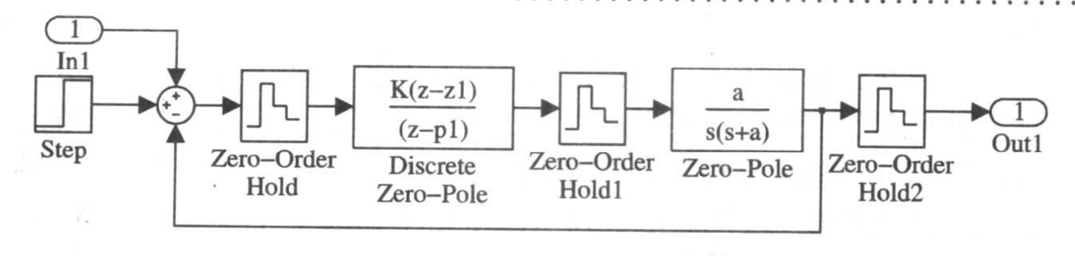


图 5-24 计算机控制系统的另一种 Simulink 表示 (文件名：c5mcomp2.mdl)

5.5.3 基于误差的最优控制器设计程序 OCD 及应用

由前面的演示可以看出，基于数值最优化技术的最优控制器设计方法不必拘泥于传统的最优控制格式，可以任意定义目标函数，故它应该比传统的最优控制有更好的应用前景。

作者总结了伺服控制的一般形式，编写了一个基于跟踪误差指标的最优控制器设计程序，依赖 MATLAB 和 Simulink 求解出真正最优的控制器参数，该程序允许用户用 Simulink 描述控制系统模型，其中控制器可以由任意形式给出，允许带有待优化的参数，并可以自动生成最优化需要的目标函数求解用的 MATLAB 函数，然后调用相应的最优化问题求解函数，求出最优控制器的参数。

最优控制器设计程序 (Optimal Controller Designer，OCD) 的调用过程为：

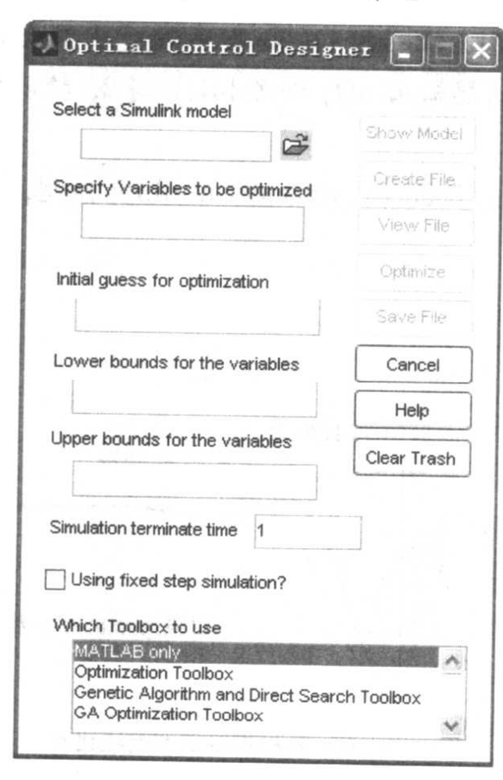


图 5-25 最优控制器设计程序界面

① 在 MATLAB 提示符下输入 `ocd`，则将得出如图 5-25 所示的程序界面，该界面允许用户利用 MATLAB/Simulink 提供的功能设计最优控制器。

② 建立一个 Simulink 仿真模型，该模型应该至少包含以下两个内容：首先应含有待优化的参数变量，这可以在框图的模块参数中直接反映出来，例如在 PI 控制器中使用 `Kp` 和 `Ki` 来表示其参数；另外，误差信号的准则需要用输出端子模块表示，例如若选择系统误差信号的 ITAE 作为目标函数，则需要将误差信号后接 ITAE 模块，并将其连接到输出端子 1。

③ 将对应的 Simulink 模型名填写到 Select a Simulink model 编辑框中。

④ 将待优化变量名填写到 Specify Variables to be optimized 编辑框中，且各个变量名之间用逗号分隔。

⑤ 另外还需估计指标收敛的时间段作为终止仿真时间，例如若选择 ITAE 指标，则理论上应该选择的终止仿真时间为 ∞，但在数值仿真时不能这样选择，且时间选择过长则将影响暂态结果，所以应该选择 ITAE 积分刚趋于平稳处的时间填写到 Simulation terminate time 栏目中去，注意，这样的参数选择可能影响寻优结果。

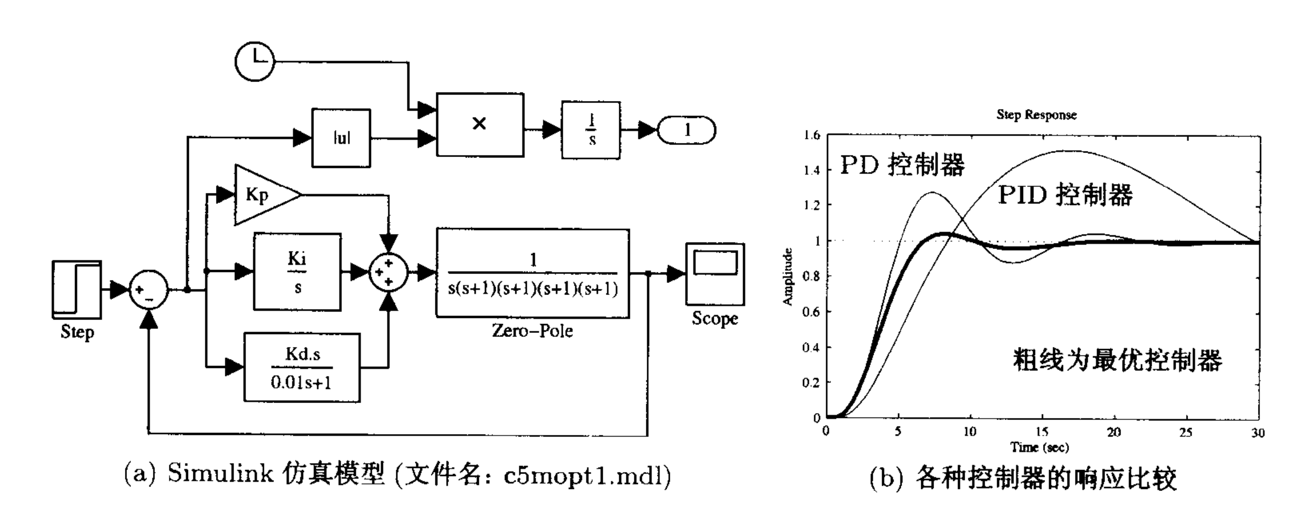
⑥ 可以单击 Create File 按钮自动生成描述目标函数的 MATLAB 文件 opt_*.m。OCD 将自动安排一个文件名来存储该目标函数，单击 Clear Trash 按钮可以删除这些暂存的目标函数文件。

⑦ 单击 Optimize 按钮将启动优化过程，对指定的参数进行寻优，在 MATLAB 工作空间中返回，变量名与上面编辑框中填写的完全一致。在实际控制器设计中，为确保能得到理想的控制器，有时需要再次单击此按钮获得更精确最优解。在实际的程序中，该按钮将根据需要自动调用 MATLAB 下的最优化函数 `fminsearch()`，`fmincon()` 或 `nonlin()` 进行参数寻优。

⑧ 本程序允许用户指定优化变量的上下界，允许用户自己选择优化参数的初值，还允许选择不同的寻优算法，并允许选择离散仿真算法等，这些都可以通过相应的编辑框和列表框直接实现。

例 5-42 受控对象的模型为 $G(s) = 1/[s(s+1)^4]$，试利用 OCD 程序设计出最优的 PID 控制器，使得系统的 ITAE 准则最小。若控制器输出 $u(t)$ 满足 $|u(t)| \leqslant 2$，控制器参数应该如何选择？

求解 如想设计最优 PID 控制器，可以建立起如图 5-26 (a) 所示的 Simulink 模型，将受控对象模型设置成 $1/[s(s+1)^4]$。



(a) Simulink 仿真模型 (文件名：c5mopt1.mdl)　　(b) 各种控制器的响应比较

图 5-26　PID 控制器仿真模型及控制效果

在 MATLAB 命令窗口输入 ocd 命令，则可以启动最优控制器设计程序，得出如图 5-25 所示的界面。在 Select a Simulink model 编辑框中填写 `c5mopt1`，在 Specify

Variables to be optimized 编辑框中填写 Kp,Ki,Kd，在 Simulation terminate time 栏目填写 30，单击 Create File 按钮，则可以自动生成目标函数的 MATLAB 如下:

```
function y=optfun_2(x)
assignin('base','Kp',x(1));
assignin('base','Ki',x(2));
assignin('base','Kd',x(3));
[t_time,x_state,y_out]=sim('c5mopt1',[0,30.000000]);
y=y_out(end);
```

其中 2~4 条程序将优化变量赋给 MATLAB 工作空间中 K_p, K_i, K_d，第 5 条语句在当前 x 向量的参数下对 Simulink 模型进行仿真，第 6 条语句将 ITAE 值赋给输出 y，完成目标函数的计算。

单击 Optimize 按钮则可以开始寻优过程。若同时打开 Simulink 模型中的示波器，则可以直接地观察寻优过程。经过寻优，可以得出使得 ITAE 指标最小的 PID 控制器为 $G_c(s) = 0.2583 + 0.0001/s + 0.7159s/(0.01s+1)$，其中积分器加权系数为 0.0001，可以忽略，故可以理解成 PD 控制器。在该控制器下系统的闭环阶跃响应如图 5-26 (b) 所示。可见，设计出的最优控制程序设计出的控制器是令人满意的。

例 5-43 最优控制程序不限于简单 PID 类控制器的设计，假设有更复杂的控制结构，比如如图 5-27 所示的串级 PI 控制器。传统的方法需要先设计内环控制器，再设计外环控制器，这里将介绍用 OCD 同时设计串级控制器的方法。

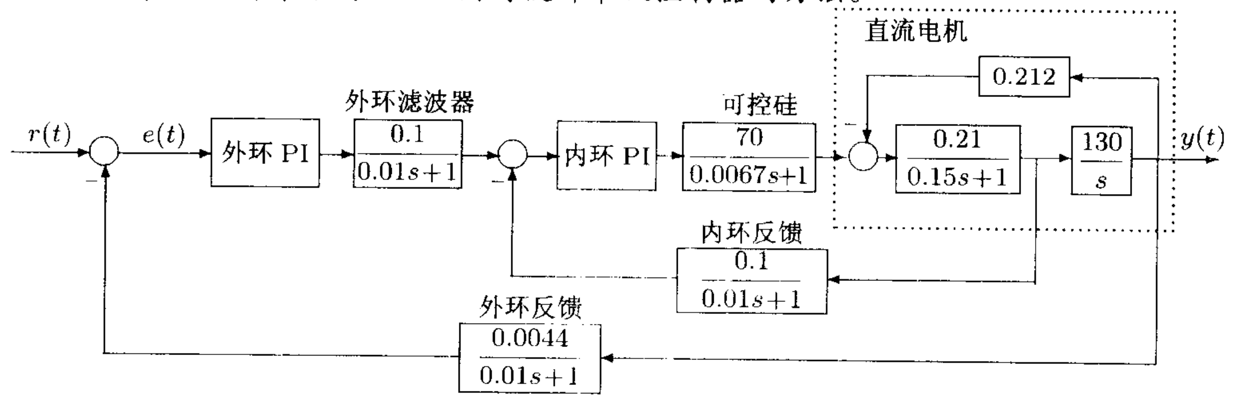


图 5-27 双闭环直流电机拖动系统框图

要解决这样的问题，需要建立起如图 5-28 所示的 Simulink 仿真模型。注意在该模型中定义了 4 个待定参数，Kp1, Ki1, Kp2, Ki2，并定义了误差的 ITAE 指标，输出到输出端子 1 上。启动 OCD，在 Select a Simulink model 编辑框中填写 c5model2.mdl，在 Specify Variables to be optimized 编辑框中填写 Kp1,Ki1,Kp2,Ki2，并在 Simulation terminate time 栏目填写终止时间 0.6，则可以单击 Create File 按钮生成描述目标函数的 MATLAB 文件，再单击 Optimize 按钮，则可以得出 ITAE 最优化设计参数为 $K_{p1} = 37.9118$，$K_{i1} = 12.1855$，$K_{p2} = 10.8489$，$K_{i2} = 0.9591$，亦即外环控制器模型

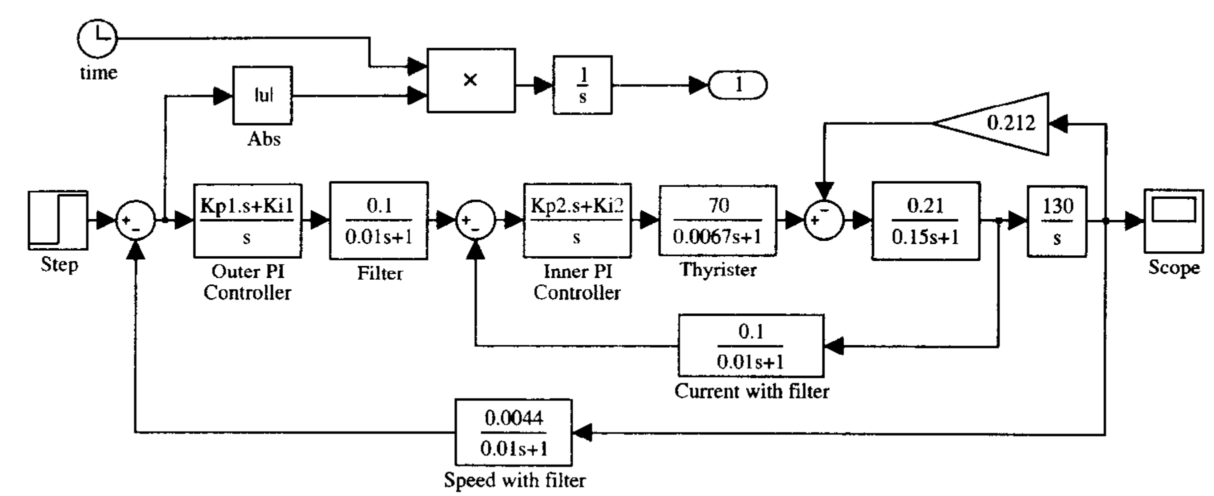


图 5-28　串级控制的 Simulink 仿真模型 (文件名：c5model2.mdl)

为 $G_{c1}(s) = 37.9118 + 12.1855/s$，内环控制器为 $G_{c2}(s) = 10.8489 + 0.9591/s$。在这些控制器下系统的阶跃响应曲线如图 5-29 所示，可见系统响应还是很理想的。

5.5.4　最优控制一般问题求解程序 RIOTS 简介

基于 MATLAB 的 RIOTS 程序 (Recursive Integration Optimal Trajectory Solver)[16]是求解一般最优控制问题的较实用的程序。假设受控对象的状态方程模型为

$$\dot{x}(t) = \boldsymbol{h}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{5-5-11}$$

根据该程序，最优控制一般问题的数学描述为[17]

$$\min_{\boldsymbol{u},\boldsymbol{x}_0 \ \text{s.t.} \begin{cases} \boldsymbol{l}_{ti}(t,\boldsymbol{x},\boldsymbol{u}) \leqslant 0 \\ \boldsymbol{g}_{ci}(t,\boldsymbol{x},\boldsymbol{u}) \leqslant 0 \\ \boldsymbol{g}_{ee}(t,\boldsymbol{x},\boldsymbol{u}) = 0 \\ \boldsymbol{u}_m(t) \leqslant \boldsymbol{u}(t) \leqslant \boldsymbol{u}_M(t) \\ \boldsymbol{x}_{0,m} \leqslant \boldsymbol{x}_0 \leqslant \boldsymbol{x}_{0,M} \end{cases}} \left[g_{\rm o}(\boldsymbol{x}_0, \boldsymbol{x}_{\rm f}) + \int_0^{t_{\rm f}} l_{\rm o}(t, \boldsymbol{x}, \boldsymbol{u}) {\rm d}t \right] \tag{5-5-12}$$

可见，很多最优控制中的目标函数，如线性二次型最优控制都可以用这样的一般模型描述。从给出的系统模型、目标函数和约束条件可以看出，这里涉及很多已知函数和数值，见表 5-4。

定维矩阵 $\boldsymbol{n}_{\rm eq}$ 需要首先给出，该矩阵是由双列的形式给出，第一列给出定维的编号，第 2 列为该编号的定维值。对应的定维向量值的定义为 $\boldsymbol{n} = [n, p, n_{\rm u}, \times, \times, n_{\rm o}, \cdots]$，其中，$n$ 为受控对象的阶次，p 为输入路数，$n_{\rm u}$ 为输入变量点数，× 表示目前保留的维数，可以给出任意值。$n_{\rm o}$ 为目标函数的个数，默认

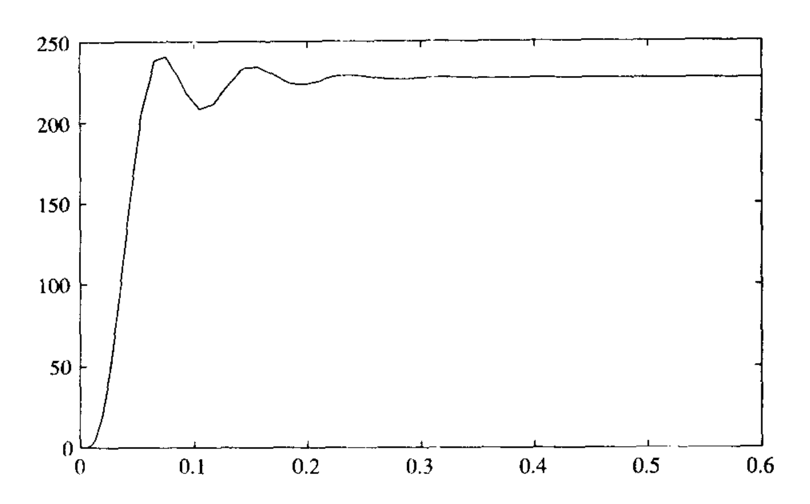


图 5-29 拖动系统最优控制阶跃响应

表 5-4 Riots 程序参数和用户函数表

函数	数学描述	调用格式	说 明
sys_init	$\boldsymbol{n}_{\rm eq}$	$\boldsymbol{n}_{\rm eq}$=sys_init	定维初始化
sys_h	$\boldsymbol{h}(t,\boldsymbol{x},\boldsymbol{u})$	y=sys_h($\boldsymbol{n}_{\rm eq}$,t,$\boldsymbol{x}$,$\boldsymbol{u}$)	受控对象模型描述
sys_g	$g_{\rm o}(\boldsymbol{x}_0,\boldsymbol{x}_{\rm f})$	$\boldsymbol{y}$=sys_g($\boldsymbol{n}_{\rm eq}$,t,$\boldsymbol{x}_0$,$\boldsymbol{x}_{\rm f}$)	终端函数
sys_l	$l(t,\boldsymbol{x},\boldsymbol{u})$	$\boldsymbol{y}$=sys_l($\boldsymbol{n}_{\rm eq}$,t,$\boldsymbol{x}$,$\boldsymbol{u}$)	动态目标函数
sys_dg	$\partial g(t,\boldsymbol{x},\boldsymbol{u})$	[$g_{\boldsymbol{x}_0}$,$g_{\boldsymbol{x}_{\rm f}}$,$g_{\rm t}$]=sys_dg($\boldsymbol{n}_{\rm eq}$,$t$,$\boldsymbol{x}_0$,$\boldsymbol{x}_{\rm f}$)	终端函数导数
sys_dl	$\partial l(t,\boldsymbol{x},\boldsymbol{u})$	[$l_{\boldsymbol{x}}$,$\boldsymbol{l}_{\boldsymbol{u}}$,$l_{\rm t}$]=sys_dl($\boldsymbol{n}_{\rm eq}$,$t$,$\boldsymbol{x}_0$,$\boldsymbol{x}_{\rm f}$)	动态目标函数导数
sys_dh	$\partial \boldsymbol{h}(t,\boldsymbol{x},\boldsymbol{u})$	[$\boldsymbol{A}_{\rm h}$,$\boldsymbol{B}_{\rm h}$]=sys_dh($\boldsymbol{n}_{\rm eq}$,t,$\boldsymbol{x}_0$,$\boldsymbol{x}_{\rm f}$)	系统函数导数

值为 1。例如，若某系统的状态变量个数为 6，输入路数为 1，目标函数的个数为 1，则定维矩阵可以写成 $\boldsymbol{n}_{\rm eq}$=[1,6; 2,1; 6,1]。

为加快最优化的速度，还可以提供这里涉及的函数导数，即

$$\boldsymbol{A}_{\rm h}(i,j)=\frac{\partial h_i(t,\boldsymbol{x},\boldsymbol{u})}{\partial x_j},\ \ \boldsymbol{B}(i,k)=\frac{\partial h_i(t,\boldsymbol{x},\boldsymbol{u})}{\partial u_k} \tag{5-5-13}$$

$$\boldsymbol{l}_{\boldsymbol{x}}(i)=\frac{\partial l(t,\boldsymbol{x},\boldsymbol{u})}{\partial x_i},\ \ \boldsymbol{l}_{\boldsymbol{u}}(k)=\frac{\partial l(t,\boldsymbol{x},\boldsymbol{u})}{\partial u_k} \tag{5-5-14}$$

$$\boldsymbol{g}_{\boldsymbol{x}_0}(i)=\frac{\partial g(t,\boldsymbol{x}_0,\boldsymbol{u})}{\partial x_i(0)},\ \ \boldsymbol{g}_{x_{\rm f}}(i)=\frac{\partial g(t,\boldsymbol{x}_{\rm f},\boldsymbol{u})}{\partial x_{\rm f}(i)} \tag{5-5-15}$$

其中，$i,j=1,2,\cdots,n,k=1,2,\cdots,p$，而它们对 t 的导数无需定义。定义了这些函数之后，就可以调用 `riots()` 函数来求解最优控制问题了。该函数的调用格

式为

$$[\boldsymbol{u},\boldsymbol{x},f]=\texttt{riots}(\boldsymbol{x}_0,\boldsymbol{u}_0,\boldsymbol{t},\boldsymbol{u}_{\rm m},\boldsymbol{u}_{\rm M},\texttt{pars},[m_{\rm i},n_{\rm v},k_{\rm d}],k_{\rm a})$$

其中，$\boldsymbol{x}_0$ 和 $\boldsymbol{u}_0$ 为初始状态和初始输入，$\boldsymbol{t}$ 为用户选定的时间向量。从计算量的角度考虑，$\boldsymbol{t}$ 向量不宜选择过多的点，在选定的点以外可以考虑用样条函数进行插值。$m_{\rm i}$ 为最大允许的迭代次数，$n_{\rm v}$ 为防止导数变化过大引入的约束，$k_{\rm d}$ 为导数约束的情况，若选择非零的 $k_{\rm d}$ 则可以略去 sys_dg(), sys_dl() 和 sys_dh() 等函数的调用。$k_{\rm a}$ 是算法选择，一般可以选择为 6。得出的 $\boldsymbol{u}$, $\boldsymbol{x}$ 为计算出的输入信号和状态变量的仿真结果，f 为计算出的目标函数值。

由于在计算时计算点选择较少，直接由 plot(t,u) 等命令绘制出的曲线较粗糙，所以需要采用 **sp_plot(t,u)** 命令绘制样条插值曲线。

例 5-44 考虑文献 [18] 中给出 Boeing 747 的着陆模型

$$\begin{cases}\dot{\boldsymbol{x}}(t)=\begin{bmatrix}-0.089 & -2.19 & 0 & 0.319 & 0 & 0\\ 0.076 & -0.217 & -0.166 & 0 & 0 & 0\\ -0.602 & 0.327 & -0.975 & 0 & 0 & 0\\ 0 & 0.15 & 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 2.19 & 0\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0 & 0.0327\\ 0.0264 & -0.151\\ 0.227 & 0.0636\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}\boldsymbol{u}(t)\\ \boldsymbol{y}(t)=[-0.089,\ 0,\ 0,\ 0,\ 0,\ 0]\boldsymbol{x}(t)+[0,\ 0.0327]\boldsymbol{u}(t)\end{cases}$$

设计的目标是使得在 $t_{\rm f}=10$ 时，状态变量 $x_6(t_{\rm f})=10$，其他状态变量收敛于 0，则可以选定最优控制的目标函数为

$$\min_{\boldsymbol{u}}=10000\left|\left|\boldsymbol{x}(t_{\rm f})-[0,0,0,0,0,10]^{\rm T}\right|\right|_2+\int_0^{t_{\rm f}}\left[100y^2(t)+\boldsymbol{u}^{\rm T}(t)\boldsymbol{u}(t)\right]{\rm d}t$$

试求出该模型的最优控制。

求解 分析该问题，可见该系统有 6 个状态变量，有两路输入信号。由于可以不提供各个导数函数，所以 sys_dg(), sys_dl() 和 sys_dh() 函数无需给出，这时需将 $k_{\rm d}$ 选择成非零整数，如选择 1。这样，可以写出下面的一些 MATLAB 函数来描述整个问题

```
function neq=sys_init(params)
neq=[1,6; 2,2; 12,6];
function xdot=sys_h(neq,t,x,u)
A=[-0.089,-2.19,0,0.319,0,0; 0.076,-0.217,-0.166,0,0,0;
    -0.602,0.327,-0.975,0,0,0; 0,0.15,1,0,0,0;
    0,1,0,0,0,0; 1,0,0,0,2.19,0];
B=[0,0.0327; 0.0264,-0.151; 0.227,0.0636; 0,0; 0,0; 0,0];
xdot=A*x+B*u;
function z=sys_l(neq,t,x,u)
C=[-0.089, 0, 0, 0, 0, 0]; D = [0, 0.0327];
ay=C*x+D*u; z=10*10*ay^2+u'*u;
```

```
function J=sys_g(neq,t,x0,xf)
J=10000*norm(xf-[0, 0, 0, 0, 0, 10]');
```

求解整个问题时，m_i 应该设置成较大的数值，如 100，这样才能够保证得出的控制收敛。使用下面的命令来求解整个最优控制问题，经 71 次迭代，可以得出最优控制信号和最优状态轨迹，如图 5-30 所示。

```
>> N=10; x0=[0; 0; 0; 0; 0; 0]; t=[0:10/N:10]; u0=zeros(2,N+3-1);
   [u,x,f] = riots(x0,u0,t,[],[],[],[100, 0, 1], 4);
   sp_plot(t,u), figure; sp_plot(t,x)
```

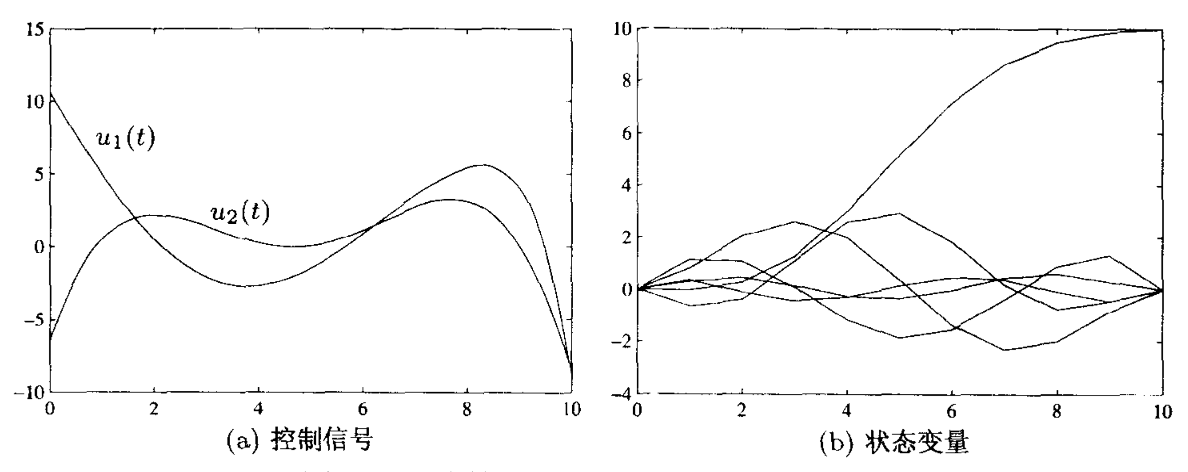


图 5-30　计算出来的最优控制与状态信号

5.5.5　参数不确定系统的最优控制器设计

假设系统的某些参数在已知区间内变化，则可以考虑采用 Minimax 最优化问题求解。Minimax 问题的一般描述方法为

$$J = \min_{\boldsymbol{x} \text{ s.t. } \begin{cases} \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B} \\ \boldsymbol{A}_{\rm eq}\boldsymbol{x} = \boldsymbol{B}_{\rm eq} \\ \boldsymbol{x}_{\rm m} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\rm M} \\ \boldsymbol{C}(\boldsymbol{x}) \leqslant \boldsymbol{0} \\ \boldsymbol{C}_{\rm eq}(\boldsymbol{x}) = \boldsymbol{0} \end{cases}} \left[\max_{F_i(\boldsymbol{x})} \boldsymbol{F}(\boldsymbol{x}) \right] \tag{5-5-16}$$

其中，目标函数是一个向量函数。该问题的物理含义是，找到一个使得一组目标函数 $\boldsymbol{F}(\boldsymbol{x})$ 最坏情况 (表示为取最大值) 的最优值最小。该问题有两种求解方法，其中一种是将内部的求取最大值部分变成一个单一的目标函数问题，另一种方法是直接调用 `fminimax()` 函数直接求解 Minimax 问题。

$$[\boldsymbol{x}, f_{\rm opt}, \texttt{flag}, \texttt{c}]=\texttt{fminimax}(\texttt{F}, \boldsymbol{x}_0, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\rm eq}, \boldsymbol{B}_{\rm eq}, \boldsymbol{x}_{\rm m}, \boldsymbol{x}_{\rm M}, \texttt{CF}, \texttt{OPT}, p_1, p_2, \cdots)$$

在控制问题中，可以考虑引入 Minimax 问题，假设对某不确定系统来说，能够寻找到某个控制器，使得该控制器对不确定系统模型中控制效果最差的系统都能达到最好的控制效果，这样就能得到对不确定系统的满意控制。

例 5-45 考虑一个不确定电机模型 $G(s)=\dfrac{9}{s(s+6\zeta)}$，其中 $\zeta\in[0,1.3]$，若控制信号 $|u(t)|\leqslant 4$，试设计出一个最优的 PD 控制器。

求解 由于受控对象含有积分器，所以在控制器中没有必要含有积分项，这样 PD 控制就能较好地控制该系统。搭建一个 PD 控制的框图，如图 5-31 所示。其中输入信号采用变阶梯信号，变化范围最大暂定为 5。所以在优化时将 u_0 参数设置成 5，仿真时设置成 1。

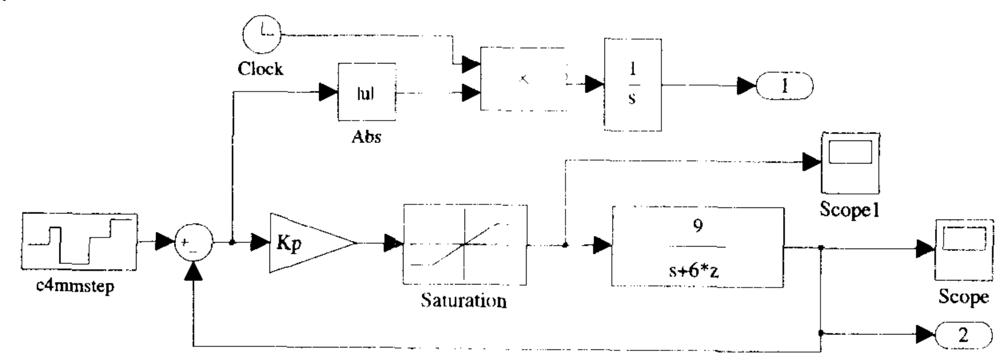


图 5-31　不确定系统比例控制仿真框图 (文件名：c5mmax.mdl)

依据该仿真模型，可以写出多目标的目标函数为

```
function y=c5minmax(x,zvec)
assignin('base','Kp',x(1)); assignin('base','Kd',x(2)); yy=[];
for z=zvec
   assignin('base','z',z);
   [t,x_sys,y_out]=sim('c5mmax',[0,1.5]); yy=[yy,y_out(end,1)];
end
y=yy;    % 返回各个不确定参数下的一组目标函数值
```

选定一组 $\zeta=[0,0.3,0.6,0.9]$，利用上面编写的目标函数，则可以求出最优的控制器参数为 $K_{\rm p}=12.8763$，$K_{\rm d}=1.5099$。

```
>> zvec=[0:0.3:0.9]; u0=5;
   x=fminimax(@c5minmax,[1;1],[],[],[],[],[],[],[],[],zvec);
```

当然，在目标函数内部求出每个 ζ 下原目标函数的最大值，则同样的问题可以调用一般最优化问题的求解方法来求解，另外，通过实践可以看出，如果能将目标函数设置成所有样本的 ITAE 值的和，则可能得出更好的控制效果。这时需要将目标函数修改成

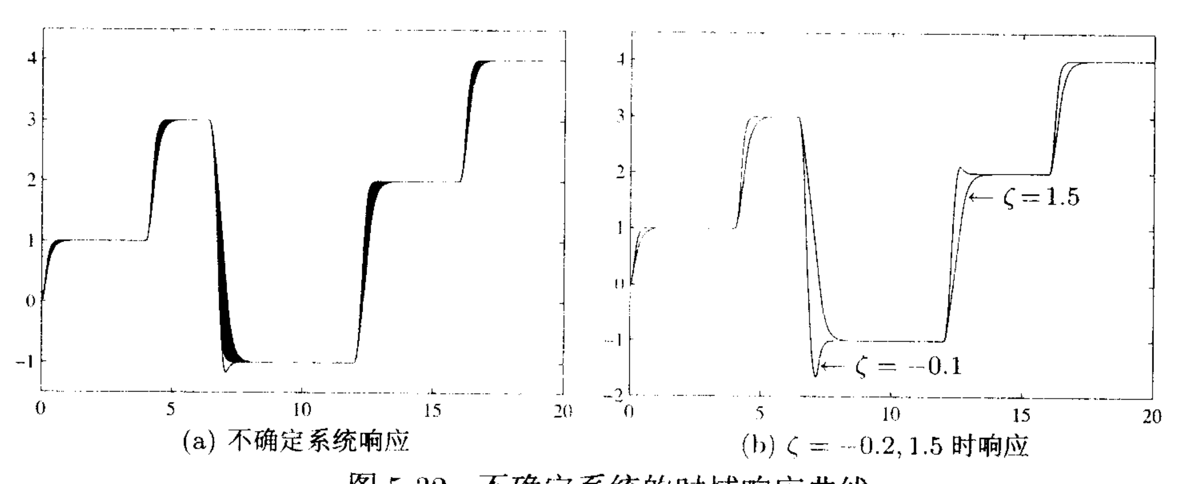
```
function y=c5minmax1(x,zvec)
assignin('base','Kp',x(1));
assignin('base','Kd',x(2));yy=[];
for z=zvec
   assignin('base','z',z);
   [t,x_sys,y_out]=sim('c5mmax',[0,1.5]); yy=[yy,y_out(end,1)];
end
y=sum(yy);  % 直接将其转换成单目标函数最优化问题
```

这时，用求解无约束最优化的方法调用 fminunc() 函数可以直接求解该问题，得出控制器参数为 $K_{\mathrm{p}} = 12.8194$，$K_{\mathrm{d}} = 2.1185$。在对本例的求解中，由这种方法求解的速度明显快于前面的 Minimax 求解函数。

```
>> x=fminunc(@c5minmax1,[1;1],[],zvec);
```

在得出的本例控制器作用下，可以对该 ζ 变化范围内一些备选的值进行仿真，并恢复多阶梯驱动信号，则得出的仿真结果如图 5-32 (a) 所示。可见，在此控制器作用下，不确定系统的时域响应曲线是令人满意的。

```
>> zvec2=[0:0.1:1.2]; u0=1; % 选择另一组 ζ 值
   for z=zvec2,
      [t,x,y]=sim('c5mmax',[0,20]); plot(t,y(:,2)), hold on
   end
```



(a) 不确定系统响应　　(b) $\zeta = -0.2, 1.5$ 时响应

图 5-32　不确定系统的时域响应曲线

再选择 $\zeta \in (0, 0.9)$ 区域外的两个值，$\zeta = -0.1$ 和 $\zeta = 1.5$，可以用同样的比例控制器进行控制，控制效果如图 5-32 (b) 所示。可见，这时的结果仍然较好，故该方法具有一定的品质鲁棒性。

```
>> z=1.5; [t,x,y]=sim('c5mmax',[0,20]); plot(t,y(:,2)), hold on;
   z=-0.1; [t,x,y]=sim('c5mmax',[0,20]); plot(t,y(:,2))
```

5.6 习题与思考题

1 求解能转换成多项式方程的联立方程，并检验得出的高精度数值解的精度。

$$\text{①}\left\{\begin{array}{l} x_1^2-x_2-1=0 \\ (x_1-2)^2+(x_2-0.5)^2-1=0 \end{array}\right., \quad \text{②}\left\{\begin{array}{l} x^2y^2-zxy-4x^2yz^2=xz^2 \\ xy^3-2yz^2=3x^3z^2+4xzy^2 \\ y^2x-7xy^2+3xz^2=x^4zy \end{array}\right.$$

2 试用图解法求解下面的一元和二元方程，并验证得出的结果。

① $f(x)=\mathrm{e}^{-(x+1)^2+\pi/2}\sin(5x+2)$, ② $f(x,y)=(x^2+y^2+xy)\mathrm{e}^{-x^2-y^2-xy}$

3 用数值求解函数求解习题 2 中方程的根，并对得出的结果进行检验。

4 试绘制下列开环系统的根轨迹曲线，并确定使单位负反馈系统稳定的 K 值范围。

① $G(s)=\dfrac{K(s+6)(s-6)}{s(s+3)(s+4-4\mathrm{j})(s+4-4\mathrm{j})}$, ② $G(s)=K\dfrac{s^2+2s+2}{s^4+s^3+14s^2+8s}$

③ $G(s)=\dfrac{1}{s(s^2/2600+s/26+1)}$, ④ $G(s)=\dfrac{800(s+1)}{s^2(s+10)(s^2+10s+50)}$

5 绘制下面状态方程系统的根轨迹，确定使单位负反馈系统稳定的 K 值范围。

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix} -1.5 & -13.5 & -13 & 0 \\ 10 & 0 & 0 & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}u(t),\ y(t)=[0,0,0,1]\boldsymbol{x}(t)$$

6 试求出使得 $\displaystyle\int_0^1 (\mathrm{e}^x-cx)^2\,\mathrm{d}x$ 取得极小值的 c 值。

7 试求解下面的无约束最优化问题。

$$\min_{\boldsymbol{x}} \begin{array}{l} 100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)+(1-x_3^2)^2+ \\ \quad 10.1\left[(x_2-1)^2+(x_4-1)^2\right]+19.8(x_2-1)(x_4-1) \end{array}$$

8 考虑 Rastrigin 函数 [19] $f(x_1,x_2)=20+x_1^2+x_2^2-10(\cos\pi x_1+\cos\pi x_2)$，试用三维曲面绘制该函数的函数值，选择初值求取该函数的最小值，并理解全局最优解和局部最优解的概念以及最优解对初值的依赖关系。

9 考虑一个简单的一元函数最优化问题求解，$f(x)=x\sin(10\pi x)+2$, $x\in(-1,2)$，试求出 $f(x)$ 取最大值时 x 的值。已知，该函数图像有很强振荡，所以采用常规最优化方法时，若初值选择不当往往会得出局部最小值。要求在本题求解中，在 $x\in(-1,2)$ 区间内随机选择 40 个初始点，按照图 5-33 中给出的流程编程，用循环的分式从每个初始点出发进行搜索，得出全局最优解。

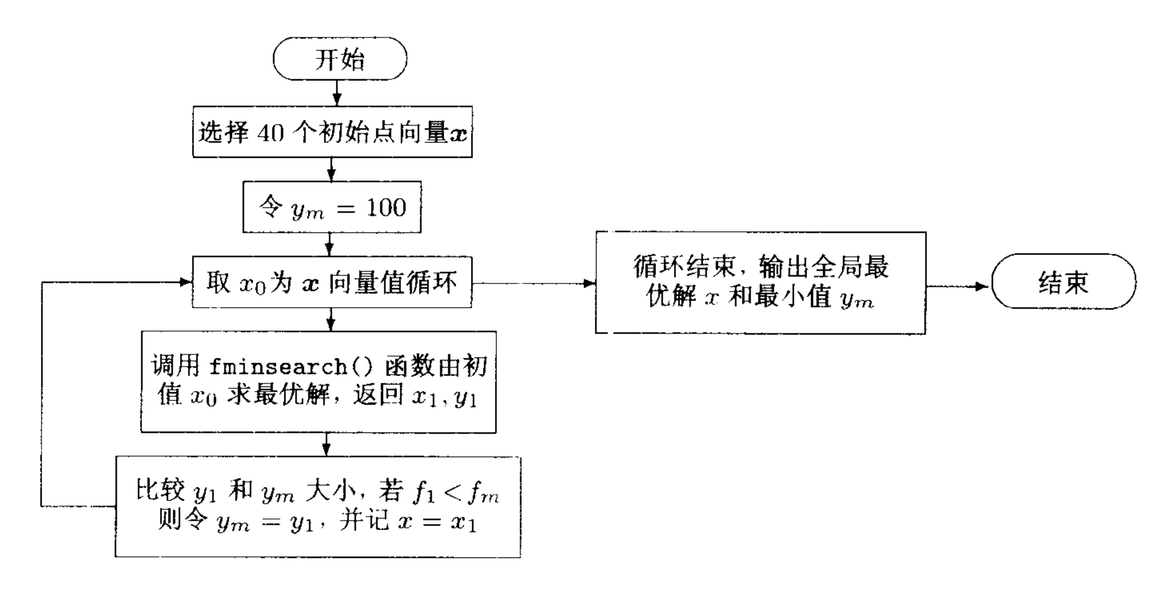


图 5-33 求取全局最优解的参考程序框图

10 如果对象模型含有纯时间延迟环节，试用最优控制器设计程序设计出 ITAE、IAE、ISE 等最优指标下的 PID 控制器，并比较控制效果。

① $G_{\mathrm{a}}(s) = \dfrac{1}{(s+1)(2s+1)}\mathrm{e}^{-s}$， ② $G_{\mathrm{b}}(s) = \dfrac{1}{(17s+1)(6s+1)}\mathrm{e}^{-30s}$

11 假设受控对象模型由延迟微分方程 $\dfrac{\mathrm{d}y(t)}{\mathrm{d}t} = \dfrac{0.2y(t-30)}{1+y^{10}(t-30)} - 0.1y(t) + u(t)$ 给出，并用 PI 控制器对系统施加控制，试将其控制转换为最优化问题进行求解，得出最优 PI 控制器参数，并绘制出系统的阶跃响应曲线。如果想减小闭环系统的超调量，则可以引入约束条件，将原始问题转换为有约束最优化问题的求解，试对该问题进行求解。

12 已知受控对象为一个时变模型 $\ddot{y}(t) + \mathrm{e}^{-0.2t}\dot{y}(t) + \mathrm{e}^{-5t}\sin(2t+6)y(t) = u(t)$，试设计一个能使得 ITAE 指标最小的 PI 控制器，并分析闭环系统的控制效果。设计最优控制器需要用有限的时间区间去近似 ITAE 的无穷积分，所以比较不同终止时间下的设计是有意义的，试分析不同终止时间下的 PI 控制器并分析效果。如果不采用 ITAE 指标而采用 IAE, ISE 等，设计出的控制器是什么？控制效果如何？

13 试为受控对象模型[20] $G(s) = \dfrac{1 + \dfrac{3\mathrm{e}^{-s}}{s+1}}{s+1}$ 设计最优控制器。

14 试用图解法求解下面的非线性规划问题，并用数值求解算法验证结果。

$$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l} x_1 - x_2 + 2 \geqslant 0 \\ -x_1^2 + x_2 - 1 \geqslant 0 \\ x_1 \geqslant 0, x_2 \geqslant 0 \end{array}\right.} (x_1^3 + x_2^2 - 4x_1 + 4)$$

15 试求解下面的线性规划问题。

① $$\min \; -3x_1+4x_2-2x_3+5x_4 \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} 4x_1-x_2+2x_3-x_4=-2\\ x_1+x_2-x_3+2x_4\leqslant 14\\ 2x_1-3x_2-x_3-x_4\geqslant -2\\ x_{1,2,3}\geqslant -1, x_4\ \text{无约束}\end{cases}$$

② $$\min \; x_6+x_7 \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} x_1+x_2+x_3+x_4=4\\ -2x_1+x_2-x_3-x_6+x_7=1\\ 3x_2+x_3+x_5+x_7=9\\ x_{1,2,\cdots,7}\geqslant 0\end{cases}$$

16 试求解下面的二次型规划问题，并用图示的形式解释结果。

① $$\min \; 2x_1^2-4x_1x_2+4x_2^2-6x_1-3x_2 \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} x_1+x_2\leqslant 3\\ 4x_1+x_2\leqslant 9\\ x_{1,2}\geqslant 0\end{cases}$$

② $$\min \; (x_1-1)^2+(x_2-2)^2 \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} -x_1+x_2=1\\ x_1+x_2\leqslant 2\\ x_{1,2}\geqslant 0\end{cases}$$

17 试求解下面的非线性规划问题。

① $$\min \; \mathrm{e}^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1) \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} x_1+x_2\leqslant 0\\ -x_1x_2+x_1+x_2\geqslant 1.5\\ x_1x_2\geqslant -10\\ -10\leqslant x_1,x_2\leqslant 10\end{cases}$$

② $$\max \; \frac{1}{2\cos x_6}\left[x_1x_2(1+x_5)+x_3x_4\left(1+\frac{31.5}{x_5}\right)\right]$$
$$\boldsymbol{x}\ \text{s.t.}\begin{cases} 0.003079x_1^3x_2^3x_5-\cos^3 x_6\geqslant 0\\ 0.1017x_3^3x_4^3-x_5^2\cos^3 x_6\geqslant 0\\ 0.09939(1+x_5)x_1^3x_2^2-\cos^2 x_6\geqslant 0\\ 0.1076(31.5+x_5)x_3^3x_4^2-x_5^2\cos^2 x_6\geqslant 0\\ x_3x_4(x_5+31.5)-x_5[2(x_1+5)\cos x_6+x_1x_2x_5]\geqslant 0\\ 0.2\leqslant x_1\leqslant 0.5,\ 14\leqslant x_2\leqslant 22,\ 0.35\leqslant x_3\leqslant 0.6\\ 16\leqslant x_4\leqslant 22,\ 5.8\leqslant x_5\leqslant 6.5,\ 0.14\leqslant x_6\leqslant 0.2618\end{cases}$$

18 求解下面的整数线性规划问题。

① $$\max \; (592x_1+381x_2+273x_3+55x_4+48x_5+37x_6+23x_7)$$
$$\boldsymbol{x}\ \text{s.t.}\begin{cases} \boldsymbol{x}\geqslant 0\\ 3534x_1+2356x_2+1767x_3+589x_4+528x_5+451x_6+304x_7\leqslant 119567\end{cases}$$

② $$\max \; (120x_1+66x_2+72x_3+58x_4+132x_5+104x_6)$$
$$\boldsymbol{x}\ \text{s.t.}\begin{cases} x_1+x_2+x_3=30\\ x_4+x_5+x_6=18\\ x_1+x_4=10\\ x_2+x_5\leqslant 18\\ x_3+x_6\geqslant 30\\ x_{1,\cdots,6}\geqslant 0\end{cases}$$

19 试求解下面的 0-1 线性规划问题，并对①、②题用穷举方法检验得出的结果。

① $$\min \; (5x_1+7x_2+10x_3+3x_4+x_5) \quad \boldsymbol{x}\ \text{s.t.}\begin{cases} x_1-x_2+5x_3+x_4-4x_5\geqslant 2\\ -2x_1+6x_2-3x_3-2x_4+2x_5\geqslant 0\\ -2x_2+2x_3-x_4-x_5\leqslant 1\\ 0\leqslant x_i\leqslant 1\end{cases}$$

② $\min\limits_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l} x_1-x_6\leqslant 0\\ x_1-x_5\leqslant 0\\ x_2-x_4\leqslant 0\\ x_2-x_5\leqslant 0\\ x_3-x_4\leqslant 0\\ x_1+x_2+x_3\leqslant 2\\ 0\leqslant x_i\leqslant 1\end{array}\right.} (-3x_1-4x_2-5x_3+4x_4+4x_5+2x_6)$, ③ $\max\limits_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}\boldsymbol{A}\boldsymbol{x}\leqslant\begin{bmatrix}600\\600\end{bmatrix}\\ 0\leqslant x_i\leqslant 1\end{array}\right.} \boldsymbol{f}^{\mathrm{T}}\boldsymbol{x}$

其中 $\boldsymbol{f}^{\mathrm{T}}=[1898,440,22507,270,14148,3100,4650,30800,615,4975,1160,4225,510,11880,479,440,490,330,110,560,24355,2885,11748,4550,750,3720,1950,10500]$,

$$\boldsymbol{A}=\begin{bmatrix}45&0&85&150&65&95&30&0&170&0&40&25&20&0&0&25&0&0&25&0&165&0&85&0&0&0&0&100\\30&20&125&5&80&25&35&73&12&15&15&40&5&10&10&12&10&9&0&20&60&40&50&36&49&40&19&150\end{bmatrix}$$

20 已知下列各个高阶系统传递函数模型，试求出最优降阶模型。

① $G(s)=\dfrac{10+3s+13s^2+3s^2}{1+s+2s^2+1.5s^3+0.5s^4}$, ② $G(s)=\dfrac{10s^3-60s^2+110s+60}{s^4+17s^2+82s^2+130s+100}$

③ $G(s)=\dfrac{1+0.4s}{1+2.283s+1.875s^2+0.7803s^3+0.125s^4+0.0083s^5}$

④ $G(z)=\dfrac{24.1467z^3-67.7944z^2+63.4768z-19.8209}{z^4-3.6193z^3+4.9124z^2-2.9633z+0.6703}$

21 考虑时间最优控制问题，假设系统的状态方程模型为[21]

$$\left\{\begin{array}{l}\dot{x}_1(t)=-x_2(t)+x_1(t)u_1(t)\\ \dot{x}_2(t)=x_1(t)-tx_3^2(t)+u_2(t)\\ \dot{x}_3(t)=-t^2x_1^3(t)-x_3(t)\end{array}\right.$$

初值为 $x_1(0)=-1,x_2(0)=1,x_3(0)=10$。如果输入 $|u_i(t)|\leqslant 1$，试求出达到控制目标 $x_3(t_{\mathrm f})=0$, $x_1^2(t_{\mathrm f})+x_2^2(t_{\mathrm f})-t_{\mathrm f}^2-1=0$ 的最小时间 $t_{\mathrm f}$ 与控制信号 $u_i(t)$。

22 考虑多变量系统模型[22] $\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\ \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$，前置矩阵 $\boldsymbol{K}=\begin{bmatrix}-0.41357 & 2.6537\\ 1.133 & -0.32569\end{bmatrix}$ 可实现初步解耦，试选择适当的目标函数，设计出系统的最优 PI 或 PID 控制器。

参考文献

[1] Chen Y Q, Xue D Y, Gu J. Analytic and numerical computation of stability bound for a class of linear delay differential equations using Lambert functions [J]. Dynamics of Continuous, Discrete and Impulsive Systems, Series B: Applications and Algorithms, 2003, Supplement:489~494

[2] Laub A. A Schur method for solving algebraic Riccati equations [J]. IEEE Transactions on Automatic Control, 1979, AC-24(6):913~921

[3] Arnold W F III, Laub A. Generalized eigenproblem algorithms and software for algebraic Riccati equations [J]. Proceedings of IEEE, 1984, 72(12):1746~1754

[4] Nelder J A, Mead R. A simplex method for function minimization [J]. Computer Journal, 1965, 7:308~313

[5] Rosenbrock H H. An automatic method for finding the greatest or least value of a function [J]. The Computer Journal. 1960, 3(3):175~184

[6] 蔡尚峰. 自动控制理论 [M]. 北京: 机械工业出版社, 1980

[7] 谢绪凯. 现代控制理论基础 [M]. 沈阳: 辽宁人民出版社, 1980

[8] Åström K J. Introduction to stochastic control theory [M]. London: Academic Press, 1970

[9] Zhou K, Doyle J. Essentials of robust control [M]. Prentice Hall, 1998

[10] O'Dwyer A. Handbook of PI and PID controller tuning rules [M]. London: Imperial College Press, 2003

[11] Davison E J. A method for simplifying linear dynamic systems [J]. IEEE Transactions on Automatic Control, 1966, AC-11:93~101

[12] Xue D, Atherton D P. A suboptimal reduction algorithm for linear systems with a time delay [J]. International Journal of Control, 1994, 60(2):181~196

[13] Hu X H. FF-Padé method of model reduction in frequency domain [J]. IEEE Transactions on Automatic Control, 1987, AC-32:243~246

[14] Gruca A, Bertrand P. Approximation of high-order systems by low-order models with delays [J]. International Journal of Control, 1978, 28:953~965

[15] Franklin G F, Powell J D, Workman M. Digital control of dynamic systems [M]. Reading MA: Addison Wesley, 3rd , 1988. (清华大学出版社有影印版)

[16] Schwartz A, Polak E, Chen Y Q. A MATLAB toolbox for solving optimal control problems [Z], 1997

[17] Schwartz A. Theory and implementation of numerical methods based on Runge-Kutta integration for solving optimal control problems [D]. Ph.D. thesis, Department of Electrical Engineering, University of California, Berkeley, 1996

[18] Bryson A E Jr. Dynamic optimization [M]. Menlo Park: Addsion Wesley, 1999

[19] Goldberg D E. Genetic algorithms in search, optimnzation and machine learning [M]. Addison-Wesley, 1989

[20] Brosilow C, Joseph B. Techniques of model-based control [M]. Englewood Cliffs: Prentice Hall, 2002

[21] Athens M, Falb P L. Optimal control – an introduction to the theory and its applications [M]. New York: McGraw-Hill Book Company, 1966

[22] Munro N. Multivariable control 1: the inverse Nyquist array design method, In: Lecture notes of SERC vacation school on control system design [M]. UMIST, Manchester, 1989

第 6 章 差分方程问题的计算机求解

类似于微分方程，差分方程在控制理论研究中也有很重要的地位，尤其是现在日益广泛应用的计算机控制技术的蓬勃发展，使得差分方程和离散系统的研究受到很多的关注。本章 6.1 节首先给出差分方程的一般概念，介绍四种线性差分方程的类型，即自回归模型、自回归各态历经模型、自回归滑动平均各态历经模型和 Box-Jenings 模型，并给出离散线性系统的表示方法。6.2 节对线性时变系统和一般非线性差分方程模型给出递推求解方法的 MATLAB 实现与 Simulink 建模与求解方法，采用 Simulink 仿真的方法理论上可以对任意的离散系统、混杂系统进行仿真分析。6.3 节介绍系统辨识的相关内容，首先介绍最小二乘辨识算法和基于 MATLAB 的系统辨识问题求解方法，并介绍基于 AIC 准则的系统阶次辨识方法，以及辨识用激励信号 —— 伪随机二进制序列信号的生成算法及其在系统辨识中的应用，最后介绍递推最小二乘辨识算法和有色噪声下的辨识方法，为后面介绍的自适应控制奠定基础。6.4 节介绍自校正控制的算法与基于 MATLAB/Simulink 的仿真方法，首先给出 Diophantine 方程求解的算法与程序，并系统介绍最优预报方法、最小方差控制算法、极点配置的控制算法等，并将辨识技术和控制技术相结合得出自校正的构造算法，介绍基于 MATLAB/Simulink 的设计和仿真方法。还将介绍一个较好的自适应控制系统设计与仿真程序。6.5 节将介绍模型预测控制方法和广义模型预测方法的设计方法和仿真方法。

6.1 差分方程与离散系统传递函数模型

6.1.1 差分方程的分类

常系数线性差分方程的一般形式为

$$y(kT)+a_1y[(k-1)T]+a_2y[(k-2)T]+\cdots+a_ny[(k-n)T]$$

$$=b_1u[(k-d)T]+b_2u[(k-d-1)T]+\cdots+b_mu[(k-d-m+1)T] \quad (6\text{-}1\text{-}1)$$

其中 T 为采样周期。和微分方程描述的连续系统类似，这里的系数 a_i 和 b_i 也是常数，所以这类系统称为线性时不变离散系统。另外，对应系统的输入信号和输出信号也可以由 $u(kT)$ 和 $y(kT)$ 表示。$u(kT)$ 为第 k 个采样周期的输入信号，$y(kT)$ 为该时刻的输出信号。为方便起见，简记 $y(t)=y(kT)$，且记 $y[(k-i)T]$ 为 $y(t-i)$，则前面的差分方程可以简记为

$$\begin{aligned}&y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)\\&\quad=b_1u(t-d)+b_2u(t-d-1)+\cdots+b_mu(t-d-m+1)\end{aligned} \quad (6\text{-}1\text{-}2)$$

后面关于离散差分方程的描述均用这样的简记表达形式。利用 Z 变换的重要性质，$\mathscr{Z}[y(t-k)]=z^{-k}\mathscr{Z}[y(t)]$，可以将前面给出的差分方程改写为

$$\left(1+a_1z^{-1}+\cdots+a_nz^{-n}\right)y(t)=\left(b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}\right)u(t-d) \quad (6\text{-}1\text{-}3)$$

其中 $y(t)$ 和 $u(t)$ 更严格地应该写成 $U(z)$ 和 $Y(z)$，这里沿用很多文献的简记方法，仍表示成 $y(t)$ 和 $u(t)$ 的格式。该方程可以进一步表示为

$$A(z^{-1})y(t)=B(z^{-1})u(t-d) \quad (6\text{-}1\text{-}4)$$

其中

$$\begin{aligned}A(z^{-1})&=1+a_1z^{-1}+\cdots+a_nz^{-n}\\B(z^{-1})&=b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}\end{aligned} \quad (6\text{-}1\text{-}5)$$

由于应用领域的不同，通常可以将差分方程描述的模型分为四类特殊的形式，本节将简要介绍各种不同的模型类型。

1. 自回归模型

自回归 (AutoRegressive，AR) 模型又称为时间序列 (time series) 模型，其数学表示为

$$y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)=0 \quad (6\text{-}1\text{-}6)$$

其中 $y(t-k)$ 为信号 y 在 k 个采样周期前的值。若想求解此差分方程，则可以采用如下的递推公式

$$y(t)=-a_1y(t-1)-a_2y(t-2)-\cdots-a_ny(t-n) \quad (6\text{-}1\text{-}7)$$

亦即，当前的 y 值可以由前 n 个采样周期的值递推得出。另外，由式 (6-1-6) 中描述的离散差分方程模型，利用 Z 变换的延迟性质 $\mathscr{Z}[y(t-k)]=z^{-k}\mathscr{Z}[y(t)]$，简记 $y(t)=\mathscr{Z}[y(t)]$，则可以将原方程表示为

$$A(z^{-1})y(t)=0,\ \text{其中}\ A(z^{-1})=1+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n} \tag{6-1-8}$$

若 $y(t)$ 为实测数据，则得出的差分方程难免会存在误差，这时 AR 型差分方程可以更一般地写成

$$y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)=\varepsilon(t) \tag{6-1-9}$$

其中若每个方程的误差 $\{\varepsilon(k)\}$ 构成的随机过程为 Gauss 过程，即 $\varepsilon(t)$ 的均值为 0，方差为 σ^2，则该信号又称为白噪声信号。

2. 自回归各态历经模型

自回归各态历经 (AutoRegressive eXogenous，ARX) 模型的数学表达式为

$$\begin{aligned}&y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)\\&\quad=b_1u(t-d)+b_2u(t-d-1)+\cdots+b_mu(t-d-m+1)+\varepsilon(t)\end{aligned} \tag{6-1-10}$$

其中，$\{\varepsilon(k)\}$ 仍为白噪声信号。这类模型可以通过 Z 变换写成

$$A(z^{-1})y(t)=B(z^{-1})u(t-d)+\varepsilon(t) \tag{6-1-11}$$

式中，$A(z^{-1})=1+a_1z^{-1}+\cdots+a_nz^{-n}$，$B(z^{-1})=b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}$。

3. 自回归滑动平均各态历经模型

考虑 ARX 差分方程模型，若方程的误差 $\epsilon(t)$ 不是白噪声，而是有色噪声信号，则可以将有色噪声 $\epsilon(t)$ 表示成白噪声信号 $\varepsilon(t)$ 的形式，即令 $\epsilon(t)=C(z^{-1})\varepsilon(t)$，这样的模型称为自回归滑动平均各态历经 (AutoRegressive Moving Average eXogenous，ARMAX) 模型，这时原差分方程模型可以改写成

$$A(z^{-1})y(t)=B(z^{-1})u(t-d)+C(z^{-1})\varepsilon(t) \tag{6-1-12}$$

式中，$A(z^{-1})=1+a_1z^{-1}+\cdots+a_nz^{-n}$，$B(z^{-1})=b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}$，$C(z^{-1})=1+c_1z^{-1}+\cdots+c_kz^{-k}$。

4. Box-Jenkins 模型

将 ARMAX 模型两端除以 $A(z^{-1})$，则可以写成

$$y(t)=\frac{B(z^{-1})}{A(z^{-1})}u(t-d)+\frac{C(z^{-1})}{A(z^{-1})}\varepsilon(t) \tag{6-1-13}$$

可见二者的分母是一致的。对这样的模型进一步进行一般化表示，即设二者的分母是不同的，则可以得出下面的差分方程模型，该模型称为 Box-Jenkins 模型

$$y(t)=\frac{B(z^{-1})}{A(z^{-1})}u(t-d)+\frac{D(z^{-1})}{C(z^{-1})}\varepsilon(t) \tag{6-1-14}$$

6.1.2 离散系统传递函数模型

2.4.3 节曾提到了差分方程和离散传递函数的概念，但并未给出二者之间的直接关系，只侧重于如何用 MATLAB 语言表示，这里将更系统地介绍离散系统的差分方程模型和离散传递函数之间的关系式。推导出和式 (6-1-10) 直接对应的离散传递函数模型为

$$H\left(z^{-1}\right)=\frac{b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n}}z^{-d} \tag{6-1-15}$$

注意，这里的表示和 2.4.3 节的表示是不同的，但是二者之间有显著的关系。如果将该传递函数分子和分母同时乘以 z^n，则可以得出 2.4.3 节的一般形式。由于这里给出的形式更适合于离散系统研究，所以本节将全面采用式 (6-1-15) 中的离散传递函数模型。

其他形式的差分方程模型可以表示成传递函数的互联形式，例如，Box-Jenkins 模型可以认为由两个传递函数模型

$$H_1(z^{-1})=\frac{B(z^{-1})}{A(z^{-1})}z^{-d},\ \ H_2(z^{-1})=\frac{D(z^{-1})}{C(z^{-1})} \tag{6-1-16}$$

分别对 $u(t)$ 和 $\varepsilon(t)$ 信号进行处理，其结果信号相加得出系统的输出信号 $y(t)$。

6.2 离散系统的求解方法

如果系统的传递函数模型由式 (6-1-15) 给出，且系统的输入 Z 变换也能写成关于 z^{-1} 的有理式，则可以将输出信号的 Z 变换看成离散传递函数模型，这样就能用脉冲响应的函数 `impulse()` 或 Simulink 的形式求解该系统。若系统由状态方程给出，还可以用递推公式直接求解该方程。本节还将演示基于 MATLAB 的非线性离散系统与连续、离散混合系统的数值求解方法。

6.2.1 线性时变系统的数值解法

线性时变系统状态方程一般可以写成

$$\begin{cases}\boldsymbol{x}(k+1)=\boldsymbol{F}(k)\boldsymbol{x}(k)+\boldsymbol{G}(k)\boldsymbol{u}(k)\\ \boldsymbol{y}(k)=\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{D}(k)\boldsymbol{u}(k)\end{cases},\ \ \boldsymbol{x}(0)=\boldsymbol{x}_0 \tag{6-2-1}$$

可见，采用递推方法，则

$$x(1) = F(0)x_0 + G(0)u(0)$$
$$x(2) = F(1)x(1) + G(1)u(1) = F(1)F(0)x_0 + F(1)G(0)u(0) + G(1)u(1)$$
$$\vdots$$

最终可以直接得出

$$\begin{aligned} x(k) &= F(k-1)F(k-2)\cdots F(0)x_0 + G(k-1)u(k-1) \\ &\quad + F(k-1)G(k-2)u(k-2) + \cdots + F(k-1)\cdots F(0)G(0)u(0) \\ &= \prod_{j=0}^{k-1} F(j)x_0 + \sum_{i=0}^{j-1}\left[\prod_{j=i+1}^{k+1} F(j)\right] G(i)u(i) \end{aligned} \tag{6-2-2}$$

若已知 $F(i), G(i)$，则可以通过上面的递推算法直接求出离散状态方程的解。从数值求解的角度看，还可以用迭代方法求解本方程，即从已知的 $x(0)$ 根据方程式 (6-2-1) 推出 $x(1)$，再由 $x(1)$ 计算 $x(2)$、$\cdots$，这样就可以得出系统在各个时刻的状态。可见，迭代法更适合计算机实现。

例 6-1 试求解离散线性时变系统[1]

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & \cos(k\pi) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \sin(k\pi/2) \\ 1 \end{bmatrix} u(k)$$

其中 $\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$，且 $u(k) = \begin{cases} 1, & k = 0, 2, 4, \cdots \\ -1, & k = 1, 3, 5, \cdots \end{cases}$。

求解 采用迭代方法，可以用下面的循环结构立即得出状态变量在各个时刻的值，如图 6-1 所示。

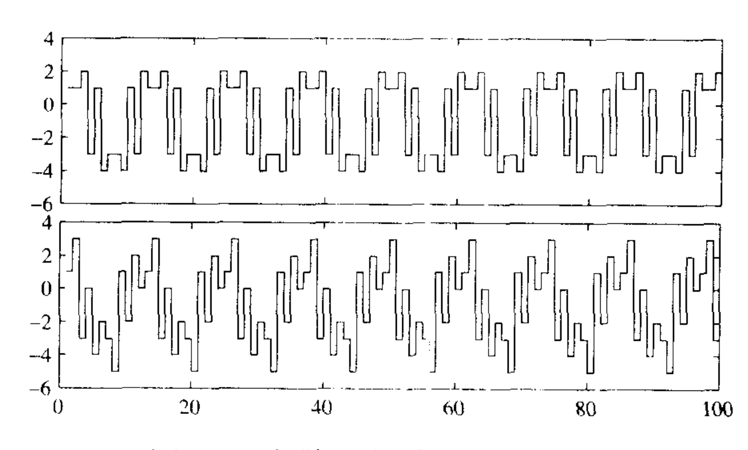


图 6-1 离散时变系统的响应曲线

```
>> x0=[1; 1]; x=x0;
   for k=0:100
```

```
    if rem(k,2)==0, u=1; else, u=-1; end
    F=[0 1; 1 cos(k*pi)]; G=[sin(k*pi/2); 1];
    x1=F*x0+G*u; x0=x1; x=[x x1];
end
subplot(211), stairs(x(1,:)), subplot(212), stairs(x(2,:))
```

6.2.2 线性时不变系统的解法

线性时不变系统有 $\boldsymbol{F}(k)=\cdots=\boldsymbol{F}(0)=\boldsymbol{F}$, $\boldsymbol{G}(k)=\cdots=\boldsymbol{G}(0)=\boldsymbol{G}$，由式 (6-2-2) 可以立即得出

$$\boldsymbol{x}(k)=\boldsymbol{F}^k\boldsymbol{x}_0+\sum_{i=0}^{k-1}\boldsymbol{F}^{k-i-1}\boldsymbol{G}\boldsymbol{u}(i) \tag{6-2-3}$$

由于计算机数学语言并不能直接求出 k 是变量形式时 $\boldsymbol{F}^k$ 的解析表达式，所以用上述的表达式无法求出状态变量的解析解，必须考虑其他的方法。

再重新考虑式 (6-2-1)，两端同时求 Z 变换，则可以得出

$$\boldsymbol{X}(z)=(z\boldsymbol{I}-\boldsymbol{F})^{-1}[z\boldsymbol{x}(0)+\boldsymbol{G}\boldsymbol{U}(z)] \tag{6-2-4}$$

这样可以推导出离散状态方程的解析解为

$$\boldsymbol{x}(k)=\mathscr{Z}^{-1}\left[(z\boldsymbol{I}-\boldsymbol{F})^{-1}z\right]\boldsymbol{x}(0)+\mathscr{Z}^{-1}\left[(z\boldsymbol{I}-\boldsymbol{F})^{-1}\boldsymbol{G}\boldsymbol{U}(z)\right] \tag{6-2-5}$$

如果已知离散系统的传递函数模型，则还可以考虑采用 2.5 节中介绍的传递函数 Z 反变换方法求解其解析解。

例 6-2 已知某离散系统的状态方程如下，试求出各个状态阶跃响应的解析解。

$$\boldsymbol{x}(k+1)=\begin{bmatrix}11/6 & -5/4 & 3/4 & -1/3\\ 1 & 0 & 0 & 0\\ 0 & 1/2 & 0 & 0\\ 0 & 0 & 1/4 & 0\end{bmatrix}\boldsymbol{x}(k)+\begin{bmatrix}4\\0\\0\\0\end{bmatrix}u(k),\ \ \boldsymbol{x}_0=\boldsymbol{0}$$

求解 直接套用下面的公式，则可以求解出状态方程的解析解为

```
>> F=sym([11/6 -5/4 3/4 -1/3; 1 0 0 0; 0 1/2 0 0; 0 0 1/4 0]);
   G=sym([4; 0; 0; 0]); syms z k; U=ztrans(sym(1));
   x=iztrans(inv(z*eye(4)-F)*G*U,z,k)
```

从而得出各个状态的解析解为

$$\boldsymbol{x}(k)=\begin{bmatrix}-12(8+k+k^2)(1/2)^k+48(1/3)^k+48\\ 24(-8+k+2k^2)(1/2)^k+144(1/3)^k+48\\ 24(-10+3k-k^2)(1/2)^k+216(1/3)^k\\ 12(-14+5k-k^2)(1/2)^k+162(1/3)^k+6\end{bmatrix}$$

6.2.3 一般非线性离散系统的求解方法

假设已知差分方程的显式形式，即

$$y(t) = f(t, y(t-1), \cdots, y(t-n), u(t), \cdots, u(t-m)) \tag{6-2-6}$$

则可以通过递推的方法直接求解该方程，得出方程的数值解。

例 6-3 假设离散非线性系统可以表示为

$$y(t) = \frac{y(t-1)^2 + 1.1y(t-2)}{1 + y(t-1)^2 + 0.2y(t-2) + 0.4y(t-3)} + 0.1u(t)$$

并假设输入信号为正弦输入 $u(t) = \sin(t)$，采样周期为 $T = 0.05$，试求解该方程的数值解。

求解 引入一个存储向量 $\boldsymbol{y}_0$，其三个分量 $y_{0,1}, y_{0,2}$ 和 $y_{0,3}$ 分别表示 $y(t-3)$, $y(t-2)$ 和 $y(t-1)$，在每一步递推后更新一次 $\boldsymbol{y}_0$ 向量。这样，用下面的循环结构就可以求解该方程，并绘制出输入信号和输出信号的曲线，如图 6-2 所示。可见，在正弦信号激励下，非线性系统的输出会产生畸变，这与线性系统响应是不同的。

```
>> y0=zeros(1,3); T=0.05; t=0:T:4*pi; u=sin(t);
   for i=1:length(t)
      y(i)=(y0(3)^2+1.1*y0(2))/(1+y0(3)^2+0.2*y0(2)+0.4*y0(1))+...
            0.1*u(i);  y0=[y0(2:3), y(i)];
   end
   plot(t,y,t,u)
```

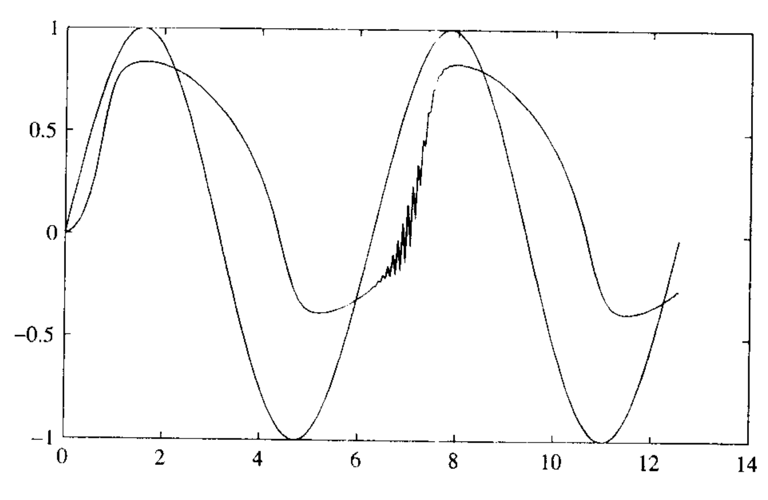


图 6-2 非线性离散系统数值解曲线

Simulink 下提供的 z^{-1} 模块可以产生一步延迟，即由 $y(t-n)$ 信号可以得出延迟信号 $y(t-n-1)$，这样可以搭建出如图 6-3 所示的仿真框图。这时得出的仿真结果与由语句得出的结果一致。

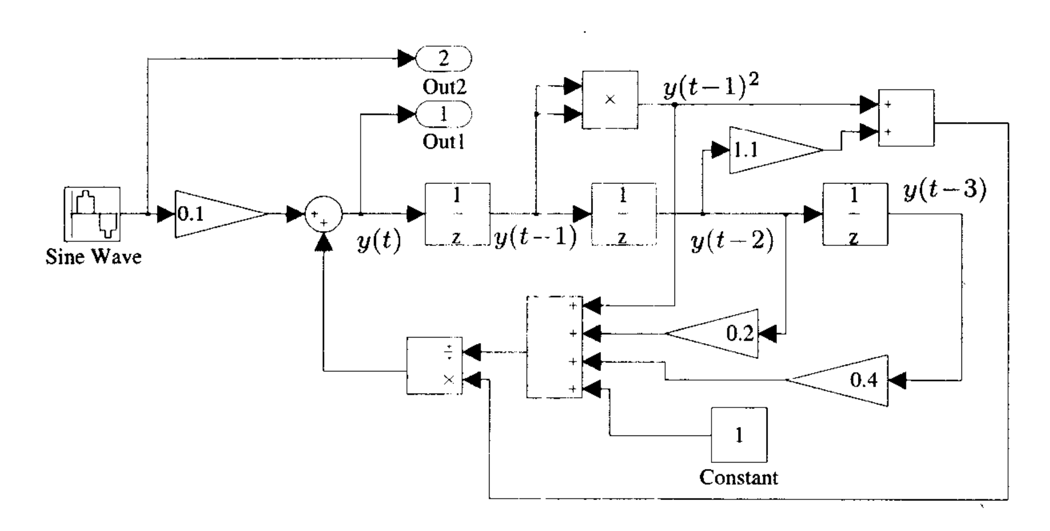


图 6-3　非线性离散系统的 Simulink 模型 (文件名：c6mnonl.mdl)

从求解过程看，似乎用 Simulink 模型求解比语句求解烦琐得多，这样的比较类似于前面关于微分方程求解。然而 Simulink 求解更能适用于一般情况，即其中含有非线性模块等特殊情况。下面给出的问题求解方法是用语句难于求解的，但用 Simulink 方法可以容易求解。

6.2.4　连续、离散混合系统的仿真方法

连续系统由微分方程表示，而离散系统由差分方程表示，在某些应用中经常会遇到整个系统中某些功能由连续系统描述，另一些功能由离散系统描述。例如，连续受控对象的计算机控制就是常见的例子，其控制器是离散的。如果不借助工具很难对其求解，我们可以借助 Simulink 对混合系统进行建模并仿真。

例 6-4　考虑例 5-41 中给出的计算机控制系统模型，其中，控制器模型是离散模型，采样周期为 T 秒，ZOH 为零阶保持器，而受控对象模型为连续模型，试对这样的连续、离散混合系统进行仿真分析。

求解　由 Simulink 可以容易地绘制出整个系统的仿真框图，如图 6-4 所示。该模型中使用了几个变量，a, T, z_1, p_1, K，其中前两个参数需要用户给定，后面 3 个参数需要由控制器模型计算。在第一个零阶保持器模块中，设置其采样周期为 T，在其他的零阶保持器和离散控制器模型中，为简单起见，采样周期均可以填写 -1，表示其采样周期继承其输入信号的采样周期，而不必每个都填写为 T。

对某受控对象 $a = 0.1$ 来说，如果选择采样周期为 $T = 0.2$ 秒，则可以用下面的语句绘制出系统阶跃响应曲线，如图 6-5 (a) 所示，其中使用阶梯图表示输出信号的采样结果。

```
>> T=0.2; a=0.1; z1=exp(-0.1*T); p1=exp(-T); K=(1-p1)/(1-z1);
   [t,x,y]=sim('c6mcompc',20);   % 启动仿真过程，得出仿真结果
```

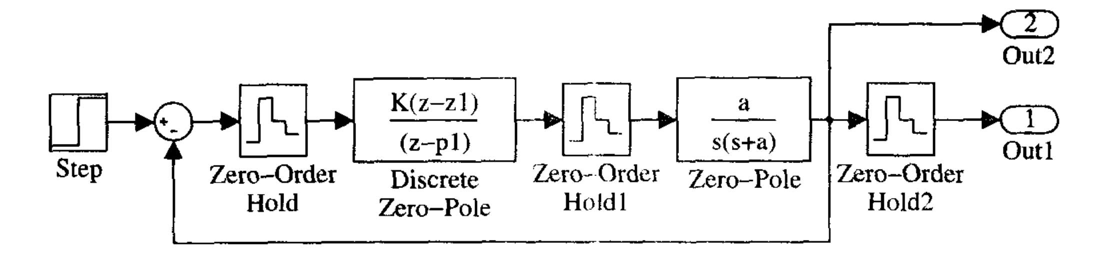
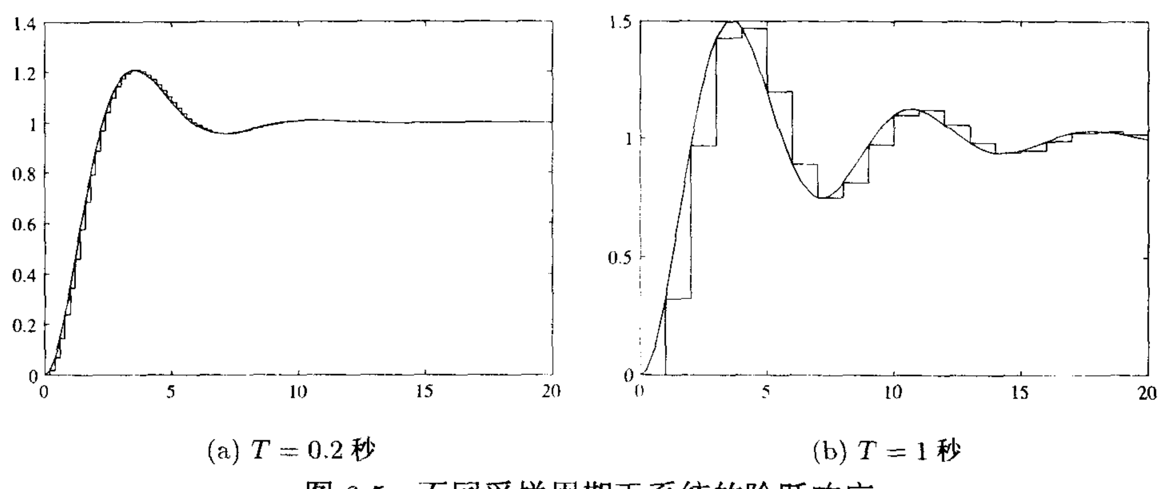


图 6-4 计算机控制系统的 Simulink 表示 (文件名：c6mcompc.mdl)

```
plot(t,y(:,2)); hold on; stairs(t,y(:,1)) % 连续、离散输出
```



(a) $T = 0.2$ 秒　　(b) $T = 1$ 秒

图 6-5 不同采样周期下系统的阶跃响应

考虑更大的采样周期 $T = 1$ 秒，可以用下面的语句绘制出系统的阶跃响应曲线，如图 6-5 (b) 所示，可见在采样周期较大时，连续信号和其采样信号相差很大。

```
>> T=1; z1=exp(-0.1*T); p1=exp(-T); K=(1-p1)/(1-z1); % 控制器参数
   [t,x,y]=sim('c6mcompc',20);  % 仿真
   plot(t,y(:,2)); hold on; stairs(t,y(:,1)) %
```

6.3 离散系统的辨识

从实测的系统输入输出数据或其他数据，用数值的手段重构系统数学模型的办法称为系统辨识。在实际应用中，可以采用许多方法从给定的系统响应数据，如时域响应中的输入和输出数据或频域响应的频率、幅值与相位数据等拟合出系统的传递函数模型，但由于这样的拟合有时解不惟一或效果较差，故一般不对连续系统数学模型进行直接辨识，而更多地对离散系统模型进行辨识。如果需要系统的连续模型，则可以通过离散模型连续化的方法，转换出系统的连续模型。本节侧重介绍离散系统的辨识方法，并给出通过选择有效的 M 序列输入信号激励系统，改进辨识精度的方法，还将介绍辨识模型阶次选择准则和基于最小二乘法的递推辨识方法。

6.3.1 离散系统的最小二乘辨识

考虑式 (6-1-10) 中给出的 ARX 差分方程模型，如果在方程中存在误差，则它对应的差分方程可以扩展为

$$\begin{aligned}&y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_ny(t-n)\\&\quad=b_1u(t-d)+b_2u(t-d-1)+\cdots+b_mu(t-d-m+1)+\varepsilon(t)\end{aligned} \tag{6-3-1}$$

其中 $\varepsilon(t)$ 为辨识的残差信号，该信号为白噪声信号。

假设已经测出了一组输入信号 $\boldsymbol{u}=[u(1),u(2),\cdots,u(M)]^{\mathrm{T}}$ 和一组输出信号 $\boldsymbol{y}=[y(1),y(2),\cdots,y(M)]^{\mathrm{T}}$，则由式 (6-3-1) 可以立即写出

$$\begin{aligned}y(1)&=-a_1y(0)-\cdots-a_ny(1-n)+b_1u(1-d)+\cdots+b_mu(2-m-d)+\varepsilon(1)\\y(2)&=-a_1y(1)-\cdots-a_ny(2-n)+b_1u(2-d)+\cdots+b_mu(3-m-d)+\varepsilon(2)\\&\quad\vdots\qquad\qquad\qquad\vdots\qquad\qquad\quad\vdots\qquad\qquad\qquad\vdots\\y(M)&=-a_1y(M-1)-\cdots-a_ny(M-n)+b_1u(M-d)\\&\qquad+\cdots+b_mu(M+1-m-d)+\varepsilon(M)\end{aligned}$$

其中 $y(t)$ 和 $u(t)$ 当 $t\leqslant 0$ 时的值均假设为零。上述方程可以写成矩阵形式

$$\boldsymbol{y}=\boldsymbol{\Phi\theta}+\boldsymbol{\varepsilon} \tag{6-3-2}$$

其中

$$\boldsymbol{\Phi}=\begin{bmatrix}y(0)&\cdots&y(1-n)&u(1-d)&\cdots&u(2-m-d)\\y(1)&\cdots&y(2-n)&u(2-d)&\cdots&u(3-m-d)\\\vdots&&\vdots&\vdots&&\vdots\\y(M-1)&\cdots&y(M-n)&u(M-d)&\cdots&u(M+1-m-d)\end{bmatrix} \tag{6-3-3}$$

$$\boldsymbol{\theta}^{\mathrm{T}}=[-a_1,-a_2,\cdots,-a_n,b_1,\cdots,b_m],\ \ \boldsymbol{\varepsilon}^{\mathrm{T}}=[\varepsilon(1),\cdots,\varepsilon(M)] \tag{6-3-4}$$

为使得残差的平方和最小，亦即 $\min\limits_{\boldsymbol{\theta}}\sum\limits_{i=1}^{M}\varepsilon^2(i)$，则可以得出待定参数 $\boldsymbol{\theta}$ 最优估计值为

$$\boldsymbol{\theta}=[\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \tag{6-3-5}$$

因为该方法是以对残差的平方和进行最小化而求解的，故这样的方法又称为最小二乘法。

MATLAB 的系统辨识工具箱中提出了各种各样的系统辨识函数，其中 ARX 模型的辨识可以由 arx() 函数加以实现。如果已知输入信号的列向量 $\boldsymbol{u}$，输出信号的列向量 $\boldsymbol{y}$，并选定了系统的分子多项式阶次 $m-1$，分母多项式阶次 n 及系统的纯滞后 d，则可以通过下面命令辨识出系统的数学模型。

T=arx([$\boldsymbol{y}$,$\boldsymbol{u}$], [n,m,d])

该函数将直接显示辨识的结果，且所得的 T 为一个结构体，其 T.$\boldsymbol{B}$ 和 T.$\boldsymbol{A}$ 分别表示辨识得出的分子和分母多项式模型。

MATLAB 的系统辨识工具箱中提供了一个 arx() 函数，可以直接用来辨识式 (6-3-1) 中的数学模型，这里将通过例子来介绍离散系统的辨识问题求解方法。

例 6-5 假设已知系统的实测输入与输出数据如表 6-1 所示，试根据这些数据辨识出系统的离散传递函数模型。

求解 假设系统分子和分母阶次均为 4，则可以根据这些数据辨识出系统的传递函数模型。首先将系统的输入输出数据输入到 MATLAB 的工作空间，然后可以直接调用 arx() 函数辨识出系统的参数。

表 6-1 已知系统的输入输出数据

t	$u(t)$	$y(t)$	t	$u(t)$	$y(t)$	t	$u(t)$	$y(t)$
0	1.4601	0	1.6	1.4483	16.411	3.2	1.056	11.871
0.1	0.8849	0	1.7	1.4335	14.336	3.3	1.4454	13.857
0.2	1.1854	8.7606	1.8	1.0282	15.746	3.4	1.0727	14.694
0.3	1.0887	13.194	1.9	1.4149	18.118	3.5	1.0349	17.866
0.4	1.413	17.41	2	0.7463	17.784	3.6	1.3769	17.654
0.5	1.3096	17.636	2.1	0.9822	18.81	3.7	1.1201	16.639
0.6	1.0651	18.763	2.2	1.3505	15.309	3.8	0.8621	17.107
0.7	0.7148	18.53	2.3	0.7078	13.7	3.9	1.2377	16.537
0.8	1.3571	17.041	2.4	0.8111	14.818	4	1.3704	14.643
0.9	1.0557	13.415	2.5	0.8622	13.235	4.1	0.7157	15.086
1	1.1923	14.454	2.6	0.8589	12.299	4.2	1.245	16.806
1.1	1.3335	14.59	2.7	1.183	11.6	4.3	1.0035	14.764
1.2	1.4374	16.11	2.8	0.9177	11.607	4.4	1.3654	15.498
1.3	1.2905	17.685	2.9	0.859	13.766	4.5	1.1022	14.679
1.4	0.841	19.498	3	0.7122	14.195	4.6	1.2675	16.655
1.5	1.0245	19.593	3.1	1.2974	13.763	4.7	1.0431	16.63

```
>> u=[1.4601,0.8849,1.1854,1.0887,1.413,1.3096,1.0651,0.7148,...
      1.3571,1.0557,1.1923,1.3335,1.4374,1.2905,0.841,1.0245,...
      1.4483,1.4335,1.0282,1.4149,0.7463,0.9822,1.3505,0.7078,...
```

```
    0.8111,0.8622,0.8589,1.183,0.9177,0.859,0.7122,1.2974,...
    1.056,1.4454,1.0727,1.0349,1.3769,1.1201,0.8621,1.2377,...
    1.3704,0.7157,1.245,1.0035,1.3654,1.1022,1.2675,1.0431]';
y=[0,0,8.7606,13.1939,17.41,17.6361,18.7627,18.5296,17.0414,...
    13.4154,14.4539,14.59,16.1104,17.6853,19.4981,19.5935,...
    16.4106,14.3359,15.7463,18.1179,17.784,18.8104,15.3086,...
    13.7004,14.8178,13.2354,12.2993,11.6001,11.6074,13.7662,...
    14.195,13.763,11.8713,13.8566,14.6944,17.8659,17.6543,...
    16.6386,17.1071,16.5373,14.643,15.0862,16.8058,14.7641,...
    15.4976,14.679,16.6552,16.6301]';
t1=arx([y,u],[4,4,1]) % 这样就可以得出分子分母均为四阶的传递函数
```

由该函数得出的结果如下所示

```
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - q^-1 + 0.25 q^-2 + 0.25 q^-3 - 0.125 q^-4
B(q) = 4.83e-008 q^-1 + 6 q^-2 - 0.5999 q^-3 - 0.1196 q^-4
Estimated using ARX
Loss function 7.09262e-010 and FPE 9.92966e-010
Sampling interval: 1
```

由显示的参数可知系统模型为

$$G\left(z^{-1}\right)=\frac{4.83\times10^{-8}z^{-1}+6z^{-2}-0.5999z^{-3}-0.1196z^{-4}}{1-z^{-1}+0.25z^{-2}+0.25z^{-3}-0.125z^{-4}}$$

亦即 $H(z)=\dfrac{4.83\times10^{-8}z^{3}+6z^{2}-0.5999z-0.1196}{z^{4}-z^{3}+0.25z^{2}+0.25z-0.125}$。

事实上，上述的数据是由例 2-37 给出的模型直接生成的，经过比较可以发现，二者还是很相近的。另外，用系统响应数据是不能辨识出系统的采样周期的，故上述系统采样周期为 1 的信息是不确切的。系统采样周期需要用表 6-1 中给出的时间信息来确定。比较正规的辨识方法是，用 iddata() 函数处理辨识用数据，再用 tf() 函数提取系统的传递函数模型

```
>> U=iddata(y,u,0.1);  % 0.1 为采样周期
   T=arx(U,[4,4,1]); % 系统辨识
   H=tf(T); G=H(1)  % 将辨识结果转换成离散传递函数模型
```

从而得出系统的传递函数模型为 $G(z)=\dfrac{4.83\times10^{-8}z^{3}+6z^{2}-0.5999z-0.1196}{z^{4}-z^{3}+0.25z^{2}+0.25z-0.125}$。

直接用 tf() 函数转换出来的传递函数模型是双输入传递函数矩阵，其第一个传

递函数是所需要的传递函数，第 2 个是从误差信号 $\varepsilon(k)$ 到输出信号的传递函数，这里可以忽略掉。

其实若不直接使用系统辨识工具箱中的 arx() 函数，也可以立即用式 (6-3-3) 和式 (6-3-5) 直接辨识系统的模型参数

```
>> Phi=[[0;y(1:end-1)] [0;0;y(1:end-2)],...
        [0;0;0; y(1:end-3)] [0;0;0;0;y(1:end-4)],...
        [0;u(1:end-1)] [0;0;u(1:end-2)],...
        [0;0;0; u(1:end-3)] [0;0;0;0;u(1:end-4)]]; % 建立 Φ
   theta=Phi\y % 辨识出结果，其中 Φ\y 即可求出最小二乘解
```

这时可以辨识结果向量为 $\boldsymbol{\theta}=[\,1,-0.25,-0.25,0.125,0,6,-0.5999,-0.1196\,]^{\mathrm{T}}$，由该向量可以改写成

```
>> Gd=tf(theta(5:8)',[1,-theta(1:4)'],'Ts',0.1) % 重建传递函数模型
```

从而辨识出系统模型为 $G(z)=\dfrac{-5.824\times10^{-7}z^3+6z^2-0.5999z-0.1196}{z^4-z^3+0.25z^2+0.25z-0.125}$。

6.3.2 辨识模型的阶次选择

从前面介绍的辨识函数可以看出，若给出了系统的阶次，则可以得出系统的辨识模型。但如何较好地选择一个合适的模型阶次呢？AIC 准则 (Akaike's information criterion) 是一种实用的判定模型阶次的准则，其定义为[2,3]

$$\mathrm{AIC}=\lg\left\{\det\left[\frac{1}{M}\sum_{i=1}^{M}\epsilon(i,\boldsymbol{\theta})\epsilon^{T}(i,\boldsymbol{\theta})\right]\right\}+\frac{k}{M} \tag{6-3-6}$$

式中 M 为实测数据的组数，$\boldsymbol{\theta}$ 为待辨识参数向量，k 为需要辨识的参数个数。可以用 MATLAB 函数 v=aic(H) 来计算辨识模型 H 的 AIC 准则的值 v，其中 H 是由 arx() 函数直接得出的 idpoly 对象。若计算出的 AIC 较小，例如小于 -20，则该误差可能对应于损失函数的 10^{-10} 级别，则这时 n,m,d 可以看成是系统合适的阶次。

例 6-6 再考虑例 6-5 中的系统辨识问题，试选择合适的系统阶次。

求解 由表 6-1 中给出的实际数据可见，在输入信号作用下，输出在第 3 步就可以得出非零的值，所以延迟的值 d 不应该超过 2。这样只需探讨 $d=0,1,2$ 几种情况，而在每一种情况下，可以用循环语句尝试各种准则的值，得出表 6-2。

```
>> U=iddata(y,u,0.1);   % 0.1 为采样周期
   for n=1:7, for m=1:7
```

```
    T=arx(U,[n,m,0]); TAic0(n,m)=aic(T);
    T=arx(U,[n,m,1]); TAic1(n,m)=aic(T);
    T=arx(U,[n,m,2]); TAic2(n,m)=aic(T);
end, end
```

表 6-2 不同阶次组合下的 AIC 准则值

n	$m=1$	2	3	4	5	6	7
			延迟步数为 $d=0$				
1	1.3487	1.3738	−0.23458	−0.63291	−1.0077	−1.5346	−2.61
2	1.2382	1.1949	−2.0995	−2.3513	−4.9058	−5.2429	−7.4246
3	1.0427	1.0427	−2.8743	−3.4523	−5.4678	−5.6186	−7.7328
4	1.0223	1.0345	−7.8505	−10.504	−20.729	−20.942	−20.946
5	1.0079	1.0287	−10.025	−13.396	−20.941	−20.982	−21.002
6	1.0293	1.0575	−13.658	−18.931	−20.944	−21.002	−21.125
7	0.98503	1.0261	−16.607	−20.701	−20.976	−20.996	−21.088
			延迟步数为 $d=1$				
1	1.484	−0.25541	−0.66303	−1.0494	−1.57	−2.6414	−3.4085
2	1.346	−2.1263	−2.3685	−4.9326	−5.2359	−7.4658	−7.6678
3	1.0658	−2.8886	−3.4758	−5.4795	−5.6407	−7.7744	−7.9316
4	1.0329	−7.8839	−10.53	−20.733	−20.973	−20.984	−20.9737
5	1.0043	−10.034	−13.406	−20.971	−21.002	−21.037	−21.0356
6	1.023	−13.694	−18.965	−20.982	−21.037	−21.148	−21.1105
7	0.9909	−16.6423	−20.7387	−21.0160	−21.0324	−21.1105	−21.1115
			延迟步数为 $d=2$				
1	−0.29215	−0.70464	−1.0849	−1.6057	−2.6827	−3.415	−3.5863
2	−2.1672	−2.4101	−4.9737	−5.2763	−7.477	−7.7083	−10.2034
3	−2.929	−3.5109	−5.5163	−5.6663	−7.8124	−7.9722	−10.5894
4	−7.9075	−10.57	−20.775	−21.013	−21.026	−21.015	−20.9850
5	−10.07	−13.438	−21.011	−21.036	−21.079	−21.077	−21.0617
6	−13.71	−18.991	−21.023	−21.078	−21.184	−21.149	−21.1646
7	−16.6792	−20.7794	−21.0574	−21.0736	−21.1488	−21.1444	−21.1393

表中，将 AIC 值低于 −20 的组合全部用阴影表示。对三种 d 的组合，可见 $(4,5,0)$, $(4,4,1)$ 和 $(4,3,2)$ 均是合适的阶次选择，它们分别对应的模型为

$$H_{4,5,0}\left(z^{-1}\right)=\frac{-2.114\times10^{-5}+3.09\times10^{-6}z^{-1}+6z^{-2}-0.5999z^{-3}-0.1196z^{-4}}{1-z^{-1}+0.25z^{-2}+0.25z^{-3}-0.125z^{-4}}$$

$$H_{4,4,1}\left(z^{-1}\right)=\frac{4.83\times10^{-8}z^{-1}+6z^{-2}-0.5999z^{-3}-0.1196z^{-4}}{1-z^{-1}+0.25z^{-2}+0.25z^{-3}-0.125z^{-4}}$$

$$H_{4,3,2}\left(z^{-1}\right)=\frac{6z^{-2}-0.5999z^{-3}-0.1196z^{-4}}{1-z^{-1}+0.25z^{-2}+0.25z^{-3}-0.125z^{-4}}$$

若选择 (5, 5, 0) 阶次组合，则可以得出如下的辨识模型

$$H_{5,5,0}\left(z^{-1}\right)=\frac{-1.074\times10^{-5}-2.343\times10^{-6}z^{-1}+6z^{-2}-0.6166z^{-3}-0.1182z^{-4}}{1-1.003z^{-1}+0.2528z^{-2}+0.2492z^{-3}-0.1256z^{-4}+0.0003231z^{-5}}$$

从得出的结果看，分母上相当于加了一个很小的 z^{-5} 项，其他项的参数与 $H_{4,5,0}\left(z^{-1}\right)$ 的分母差不多，所以在实际辨识中没有必要选择一个高的阶次。事实上，$H_{5,5,0}\left(z^{-1}\right)$ 的 AIC 值和 $H_{4,5,0}\left(z^{-1}\right)$ 相比没有显著改善，所以应该采用一个较低的阶次组合。

6.3.3 离散系统辨识信号的生成

从前面给出的例子可以看出，辨识信号产生的方式是：先产生一组 48 个输入信号，用该信号激励原始的传递函数模型，则可以得出输出信号。利用这些信号进行辨识，就可以辨识出系统的离散传递函数模型。然而，这样辨识的结果有一定的偏差。

伪随机二进制序列 (pseudo-random binary sequence，PRBS) 信号是用于线性系统辨识的很重要一类信号，以后将演示 PRBS 信号在辨识中的应用。PRBS 信号的生成方式较特殊，可以按照图 6-6 中移位寄存器的方式生成，每一个时钟脉冲进行一步向右移位。左面第一个寄存器 c_1 的值由 c_k 和 c_n 的“和”更新，即 $c_1=c_k\oplus c_n$。若 c_k、c_n 相同，则和为 0，相异则和为 1。如果想避免逻辑运算，可以将 c_1 表示成 c_k+c_n 的模二余数。这里随着寄存器个数 n 的不同，c_k 的选择也不同，具体选择可以参考表 6-3。

图 6-6 PRBS 信号生成的移位寄存器表示

表 6-3 PRBS 信号的寄存器表

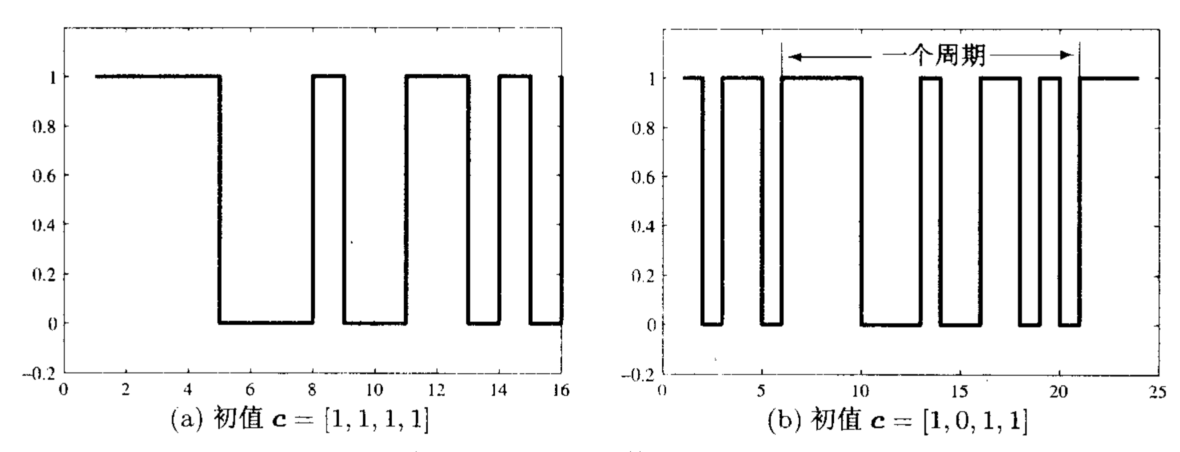
n	位数	寄存器 c_k
4	15	1 或 3
5	31	2 或 3
6	63	5
7	127	4
8	255	(2,3,4) 或 (4,5,6)
9	511	4 或 5
10	1023	7

例 6-7 试选择 4 个寄存器生成 PRBS 信号。

求解 可以令 c_1, c_2, c_3, c_4 寄存器的初值均为 1，且选 $c_k=3$，则这样由下面的语句即可以生成 15 位的 PRBS 信号，如图 6-7 (a) 所示。

```
>> M=[1 1 1 1]; c=M; k=3; n=4;
```

```
for i=1:12, c1=rem(c(k)+c(n),2); c=[c1,c(1:n-1)]; M=[M,c1]; end
stairs(M), ylim([-0.2,1.2])
```



(a) 初值 $c=[1,1,1,1]$　(b) 初值 $c=[1,0,1,1]$

图 6-7 PRBS 信号波形图

若寄存器的参数值不是 $c=[1,1,1,1]$，也可以用下面语句生成所需的 PRBS 信号，如图 6-7 (b) 所示。可见，图 6-7 (a) 的信号是该信号的一个部分。由下面语句可以生成周期为 16 的更多输入序列值，如图 6-7 (b) 所示。

```
>> M=[0 0 1 1]; c=M; k=3; n=4;
   for i=1:20, c1=rem(c(k)+c(n),2); c=[c1,c(1:n-1)]; M=[M,c1]; end
   stairs(M), ylim([-0.2,1.2])
```

利用系统辨识工具箱中的辨识信号生成函数 **u=idinput(*N*,'prbs')** 也可以生成 PRBS 信号，其中序列长度 $N=2^n-1$，n 为整数。本节将通过例子演示 PRBS 信号的生成及其在系统辨识中的应用。

例 6-8 试生成周期为 63 的 PRBS 时间序列并绘制其图形。

求解 若想生成一组 63 个点的数据，则可以通过如下的命令直接产生

```
>> u=idinput(63,'PRBS'); t=[0:.1:6.2]'; % 产生 PRBS 序列
   stairs(u), xlim([0,63]), ylim([-1.1 1.1]) % PRBS 曲线
```

得出的输入信号如图 6-8 所示。注意，这时 PRBS 曲线不能用 plot() 函数绘制。

例 6-9 利用长度为 31 的 PRBS 输入信号就可以按照例 2-37 中的方法计算出输出信号，试比较辨识效果。

求解 由这样的输入、输出数据可以直接辨识出系统的离散传递函数模型

```
>> num=[6 -0.6 -0.12]; den=[1 -1 0.25 0.25 -0.125];
   G=tf(num,den,'Ts',0.1);
   y=lsim(G,u,t); % 由离散系统模型计算系统的输出信号
   y=1e-4*round(1e4*y);  % 保留小数点后 4 位
```

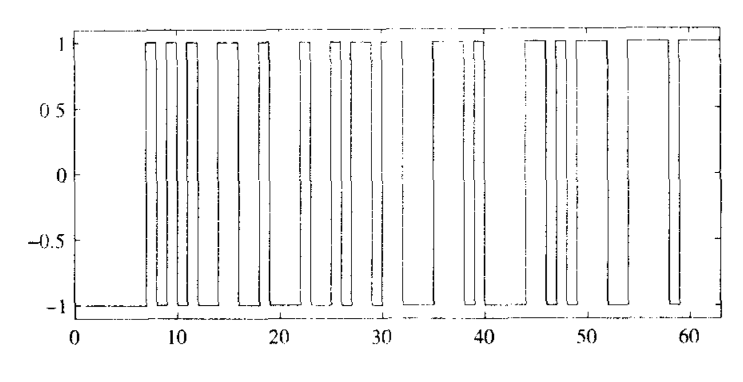


图 6-8 PRBS 序列

```
T1=arx([y,u],[4 4 1]) % 辨识系统模型, 注意看后面说明
```

得出系统的辨识模型为 $G(z)=\dfrac{-2.828\times10^{-14}z^3+6z^2-0.6z-0.12}{z^4-z^3+0.25z^2+0.25z-0.125}$，系统辨识误差达到 10^{-30} 左右。

可以看出，这样得出的系统传递函数模型与原始系统的模型完全一致。从这个例子可以看出，虽然采用的输入、输出组数比例 6-5 中少，但辨识的精度却大大高于该例中的结果，这就是选择了 PRBS 信号作为辨识输入信号的缘故。

连续系统辨识也存在各种各样的算法，例如 Levy 提出的基于频域响应拟合的辨识方法[4]，但由于频域响应拟合的非惟一性，有时辨识结果不是很理想[5]，所以可以采用间接的方法，首先辨识出离散传递函数模型，然后用连续化的方法再转化成所需的连续系统传递函数模型。

例 6-10 假设系统的传递函数模型为 $G(s)=\dfrac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$，并假设系统的采样周期为 $T=0.1$ 秒，试用 PRBS 信号和正弦信号分别激励系统，得出输入输出数据，并依照数据辨识系统模型，试比较辨识效果。

求解 用 PRBS 信号激励该系统模型，则可以用下面的语句计算出系统的输出信号

```
>> G=tf([1,7,11,5],[1,7,21,37,30]);  % 原始系统模型
   t=[0:.2:6]'; u=idinput(31,'PRBS');  % 生成 PRBS 信号
   y=lsim(G,u,t);  % 计算系统输出信号
   U=arx([y u],[4 4 1]);  % 辨识离散系统传递函数模型
   G1=tf(U); G1=G1(1); G1.Ts=0.2; G2=d2c(G1) % 连续化
```

这样可以精确地辨识出系统的传递函数模型为 $G(s)=\dfrac{s^3+7s^2+11s+5}{s^4+7s^3+21s^2+37s+30}$。

可见，这样得出的辨识模型精度还是较高的。如果不采用 PRBS 信号作为输入，而采用 81 个点的正弦信号，则仍然可以辨识出系统的离散模型，并经过连续化得出如下的结果

```
>> t=[0:.1:8]'; u=sin(t);  % 生成正弦输入信号
   y=lsim(G,u,t);  % 计算系统输出信号
   U=arx([y u],[4 4 1]);  % 辨识离散系统传递函数模型
   G1=tf(U); G1=G1(1); G1.Ts=0.1; G2=d2c(G1) % 连续化
```

这样可以辨识出系统的模型为 $G(s)=\dfrac{0.01361s^3-0.06793s^2+9.897s-2.564}{s^4+7s^3+21s^2+37s+30}$。

虽然使用正弦信号的仿真点更多了，但由于未采用有效的输入激励信号，所以得出了不准确的辨识结果。从这个例子可以看出，PRBS 信号在线性系统辨识中还是很重要的。

6.3.4 离散系统的递推辨识

如果采用前面的离散系统辨识方法，则需要首先进行输入、输出数据采集，然后构造矩阵方程，并求解该方程从而一次性地辨识出系统的参数。显然，这样一次性的辨识方法不适合于实时控制的需要。在实时控制中辨识问题经常采用递推最小二乘算法来解决。假设已知实测系统的输入输出值及其以往值 $u(t),u(t-1),\cdots,\ y(t),y(t-1),\cdots$，则可以通过这些数据用递推的方法辨识出系统的数学模型，该递推辨识算法的步骤如下。

① 选定线性系统的离散差分方程

$$\begin{aligned}&y(t+1)+a_1y(t)+\cdots+a_ny(t-n+1)\\&\quad=b_1u(t-d)+b_2u(t-d-1)+\cdots+b_mu(t-d-m+1)\end{aligned}\tag{6-3-7}$$

系统对应的离散传递函数模型为

$$G\left(z^{-1}\right)=\frac{b_1+b_2z^{-1}+\cdots+b_mz^{-m+1}}{1+a_1z^{-1}+\cdots+a_{n-1}z^{-n+1}+a_nz^{-n}}z^{-d}\tag{6-3-8}$$

其中 $m-1,n$ 分别为分子和分母多项式阶次，d 为时间延迟步数，这些数值应该为事先选定的。系统辨识的目的是通过输入和输出数据及其以往值估计出系统的参数 $a_1,\cdots,a_n,b_1,\cdots,b_m$，则可以定义出待辨识的参数向量为

$$\boldsymbol{\theta}^{\mathrm{T}}=[a_1,\cdots,a_n,b_1,\cdots,b_m]\tag{6-3-9}$$

② 选择递推辨识参数初值 $\boldsymbol{\theta}_0$ 和加权矩阵 $\boldsymbol{P}(0)$。一般情况下可以选择加权矩阵的初值为 $\boldsymbol{P}(0)=\alpha^2\boldsymbol{I}$，其中 α 可以取很大的常数[6]，且 $\boldsymbol{I}$ 为单位矩阵。

③ 定义输入输出数据向量

$$\boldsymbol{\psi}^{\mathrm{T}}(t+1)=[-y(t),\cdots,-y(t-n+1),u(t-d),\cdots,u(t-d-m+1)]\tag{6-3-10}$$

④ 由下面的递推式子可以实时地辨识出系统模型的参数[6]

$$K(t+1) = \frac{P(t)\psi(t+1)}{\lambda+\psi^{\mathrm{T}}(t+1)P(t)\psi(t+1)} \tag{6-3-11}$$

$$P(t+1) = \frac{1}{\lambda}\Big[P(t)-K(t+1)\psi^{\mathrm{T}}(t+1)P(t)\Big] \tag{6-3-12}$$

$$\theta(t+1) = \theta(t)+K(t+1)\Big[y(t+1)-\psi^{\mathrm{T}}(t+1)\theta(t)\Big] \tag{6-3-13}$$

其中 $K(t+1)$ 为中间变量，λ 称为遗忘因子，且 $0<\lambda\leqslant 1$，其值表示对过去数据的记忆程度。若 $\lambda=1$ 则表示同等处理以往数据。

⑤ 设 $t=t+1$，则转向步骤③进行下一步辨识，用这样的方法可以全程动态地对系统进行参数辨识。

上述的算法可以用 MATLAB 语言简单地实现，但考虑到该方法主要用于在线控制，更适合用 Simulink 中的 S-函数形式设计成模块。

首先分析系统辨识模块所需的输入信号和输出信号。系统辨识需要已知待辨识系统的输入信号 $u(t)$ 和输出信号 $y(t)$，可以将其作为模块的输入信号，记 $\boldsymbol{u}(t)=[u(t),y(t)]^{\mathrm{T}}$，模块的输出信号应该为系统的参数向量 $\hat{\boldsymbol{\theta}}(t)$。再比较辨识算法，式 (6-3-10) 要求已知 $u(t),y(t)$ 和其以往值，故需要按照图 6-9 中给出的形式搭建辨识模块，由单步延迟环节求出 $u(t)$ 和 $y(t)$ 前几个时刻的值。

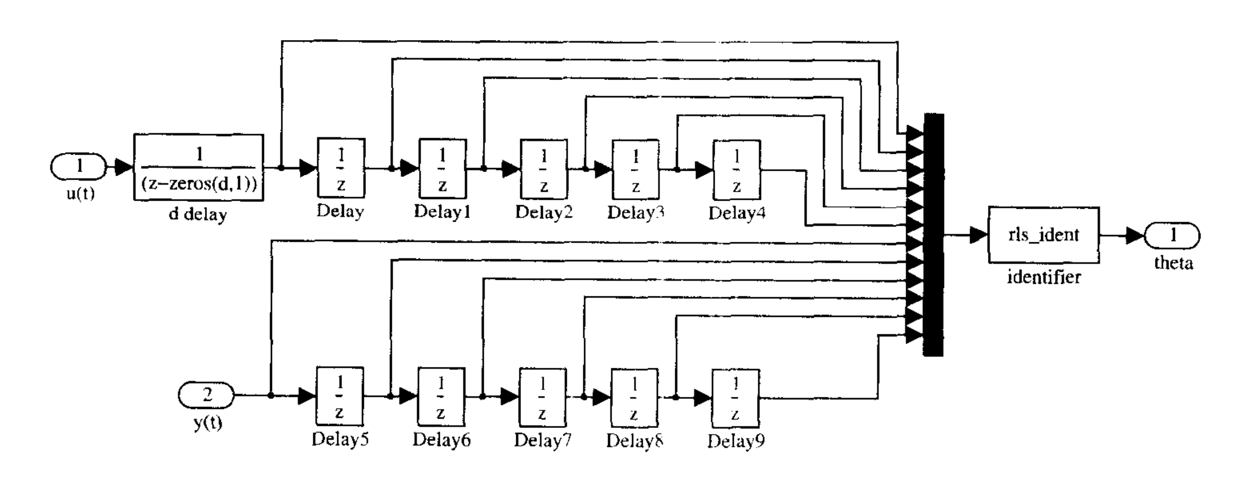


图 6-9 最小二乘递推辨识 Simulink 模块框图

在 $u(t)$ 信号的后面还加了一个可变延迟的模块，用以实现输入信号 $u(t)$ 的纯延迟，即 $u(t-d)$，这样，用给出的各个延迟环节可以获得 $u(t-d),\cdots,u(t-d-5),y(t),\cdots,y(t-5)$ 信号，所构造出模块的阶次不能超过五阶，其实这对实际系统来说应该足够了。有了这些信号，就可以定义 S-函数的输入信号了，可以将其信号定义为 $\boldsymbol{U}(t)=[u(t-d),\cdots,u(t-d-5),y(t),\cdots,y(t-5)]^{\mathrm{T}}$，亦即有 12 路输入信号。需要辨识的参数为式 (6-3-9) 中定义的 $\boldsymbol{\theta}$，故输出的路数为 $n+m+1$。其实整个模块的输出信号亦为 $\boldsymbol{\theta}$。

由于本模块中所有信号均为离散的，所以不存在连续状态变量。再分析式 (6-3-11)、式 (6-3-12) 和式 (6-3-13) 中给出的离散状态更新表达式。从给出的表达式可见，$\boldsymbol{K}(t+1)$ 的值和 $\boldsymbol{K}(t)$ 无关，故该变量为中间变量，而不是状态变量，而 $\boldsymbol{P}(t+1)$ 和 $\hat{\boldsymbol{\theta}}(t+1)$ 都应该选择为状态变量，令状态变量向量 $\boldsymbol{x}_1=\hat{\boldsymbol{\theta}}(t+1)$，$\boldsymbol{x}_2=\boldsymbol{P}(t+1)$，则可见该系统应该有状态变量 $(r+m+1)(r+m+2)$ 个。构造出状态变量 $\boldsymbol{x}^{\mathrm{T}}=[\boldsymbol{x}_1^{\mathrm{T}},\boldsymbol{x}_2^{\mathrm{T}}]$，这样可以写出如下的 S-函数来描述该模块。

```
function [sys,x0,str,ts]=rls_ident(t,x,u,flag,m,n,P0,lam)
switch flag,
   case 0 % 初始化
      [sys,x0,str,ts] = mdlInitializeSizes(m,n,P0);
   case 2  % 离散状态更新
      sys=mdlUpdate(t,x,u,m,n,lam);
   case 3 % 计算输出量，亦即控制率和权值
      sys=x(1:n+m+1); % 前 M = n + m + 1 个状态为参数向量 θ̂
   case {1, 4, 9}, sys = []; % 未定义的 flag 值
   otherwise, error(['Unhandled flag = ',num2str(flag)]);
end;
%    初始化程序
function [sys,x0,str,ts] = mdlInitializeSizes(m,n,P0)
sizes = simsizes; % 读入系统变量的默认值
sizes.NumContStates = 0; % 没有连续状态
sizes.NumDiscStates = (r+m+1)*(r+m+2); % 参数及 P 矩阵参数
sizes.NumOutputs=n+m+1; % 设置 n+m+1 路输出，即受控对象待辨识参数
sizes.NumInputs = 12; % 设置 12 路输入，输入、输出信号及其以往值
sizes.DirFeedthrough = 0; % 输入信号直接不在输出中反映出来
sizes.NumSampleTimes = 1; % 单采样速率系统
sys = simsizes(sizes); % 设置系统模型变量
x0 = [zeros(n+m+1,1); P0(:)]; % 初始状态变量（权值），设置成随机数
str = []; ts = [-1 0]; % 继承输入信号的采样周期
%    离散状态更新函数
function sys = mdlUpdate(t,x,u,m,n,lam)
psi=[-u(8:n+7); u(2:2+m)]; PN=reshape(x(n+m+2:end),n+m+1,n+m+1);
K=PN*psi/(lam+psi'*PN*psi); PN=(PN-K*psi'*PN)/lam;
sys=[x(1:n+m+1)+K*(u(7)-psi'*x(1:n+m+1)); PN(:)];
```

例 6-11 假设已知系统采样周期为 $T=0.05$ 秒，且受控对象模型为分段函数，其前 100 个采样周期的模型为 $H_1(z)=\dfrac{-z+0.3}{z^2+1.5z+0.9}$，而其后的模型变化为 $H_2(z)=\dfrac{0.72z+0.6}{z^2-0.5z-0.3}$，试用递推的方法辨识系统的模型参数。

求解 可以由分段函数的形式搭建受控对象模型，其中用一个开关来表示两个分段子模型的条件。由该系统的输入输出信号可以驱动一个递推辨识模块，则该系统参数辨识的仿真模型由图 6-10 给出。

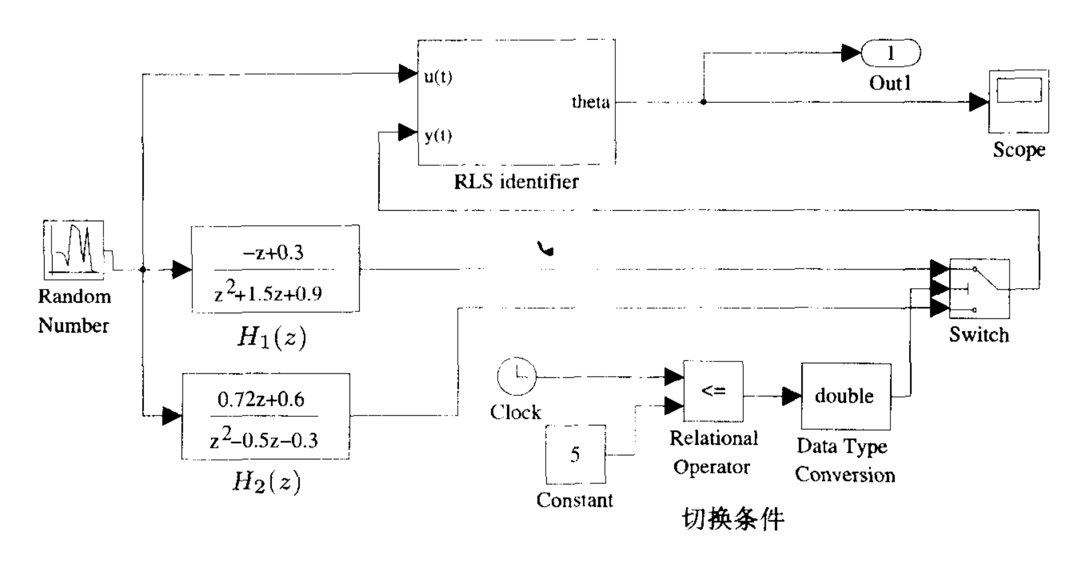


图 6-10 分段受控对象模型的辨识框图 (文件名：c6mid1.mdl)

令该模块 $\alpha = 1000$，$\lambda = 1$，并设定 $m = 1, n = 2, d = 0$，参数的初值为零向量，启动仿真过程，则可以得出系统的辨识结果，如图 6-11 (a) 所示。

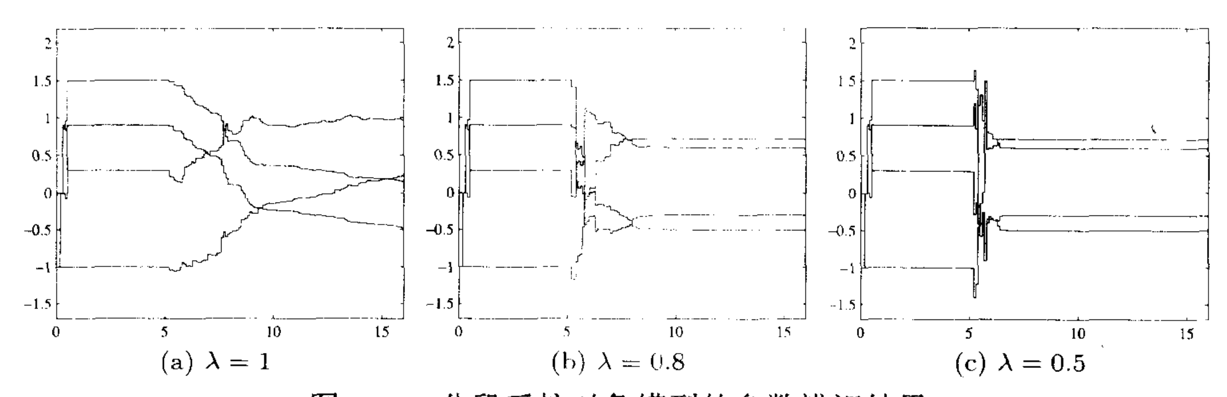


图 6-11 分段受控对象模型的参数辨识结果

由辨识结果可见，初始辨识结果是理想的，辨识参数可以立即收敛到真值，然而在模型发生突变后，由于前一个模型的输入、输出数据一直对系统辨识的结果有影响，所以模型的真值就不能准确辨识出来了。考虑到遗忘因子 λ 的作用，如果选择一个小于 1 的数值，则前一个模型的影响会逐渐消失，有助于辨识从准确的参数。图 6-11 (b) 和 (c) 分别给出了 $\lambda = 0.8$ 和 $\lambda = 0.5$ 时的系统参数，可见，这样得出的辨识结果在模型突变后仍能很快辨识出所需参数。

考虑到辨识参数波动的影响，遗忘因子 λ 的值不宜过小。

6.3.5 带有有色噪声的离散系统辨识方法

式 (6-3-1) 中曾经假设若噪声信号 $\varepsilon(t)$ 为白噪声信号，则对应的差分方程称为 ARX 模型。对该模型辨识出 $A(z^{-1})$ 和 $B(z^{-1})$ 两个多项式即可。然而如果噪

声信号 $\varepsilon(t)$ 不是白噪声信号，这样的模型应该属于 ARMX 模型，所以除了辨识出 $A(z^{-1})$ 和 $B(z^{-1})$ 两个多项式外，还需要 $C(z^{-1})$ 多项式。

在式 (6-3-2) 给出的算法中，由于 $y(t)$ 和 $u(t)$ 是可测信号，可以容易地构造 $\boldsymbol{\Phi}(t)$ 矩阵，再通过最小二乘法辨识出系统的模型参数。而对 ARMAX 模型来说，由于白噪声信号 $\varepsilon(t)$ 是不可测的，所以不能用前面的方法辨识 $C(z^{-1})$ 多项式，只能采用新的方法[3]。

采用 MATLAB 系统辨识工具箱中给出的 `armax()` 函数则可以直接辨识 ARMAX 系统的数学模型。该函数的调用格式很直观

```
H=armax([y,u],[n,m,k,d])
```

其中 $\boldsymbol{y}$, $\boldsymbol{u}$ 的定义和前面一致，表示系统的输入输出信号实测值，n, m, k 分别为多项式 $A(z^{-1})$, $B(z^{-1})$ 和 $C(z^{-1})$ 的预期阶次，d 为系统的延迟时间。和 ARX 系统辨识相似，若选择了这几个参数，则可以直接得出系统的辨识模型 H。这里得出的 H 是辨识模型对象，其中包括这几个多项式的信息和其他信息，可以通过 `tf()` 函数提取具体的模型参数。

例 6-12 假设某系统的输入和输出数据由表 6-4 给出，试辨识系统的 ARMAX 模型。

表 6-4　实验输入输出数据

u_i	y_i	u_i	y_i	u_i	y_i	u_i	y_i
−1	0	−1	−1.0792	−1	−1.1398	1	−1.0962
−1	−1.0433	1	−0.82783	−1	−0.81099	1	0.75915
−1	−1.0493	−1	1.1727	1	−0.77641	1	1.1644
−1	−0.7582	−1	−1.1212	1	1.2003	−1	0.55121
−1	−0.90227	−1	−1.066	1	0.97471	−1	−0.8946
−1	−0.96759	1	−0.68287	−1	0.88602	1	−0.99165
1	−0.71735	−1	0.88435	1	−1.2276	1	1.4171
−1	1.1856	−1	−0.88707	−1	1.0991	1	0.84192
1	−1.0971	1	−0.93207	−1	−1.0116	1	0.77319
−1	1.031	−1	1.1745	−1	−1.2223	−1	0.96745
1	−0.92971	1	−1.0416	−1	−0.60889	1	−1.2756
−1	0.87743	1	1.1169	1	−1.1249	1	1.1331
−1	−0.79738	−1	0.85354	1	1.3905	1	0.97928
1	−1.2235	1	1.3582	−1	0.73242	1	0.74469
1	1.5932	1	1.258	1	−1.0738	1	0.92557
−1	0.68265	−1	0.88018	−1	1.0923		

求解　仿照前面的辨识例子，可以选择不同的阶次组合。对本例来说，如果选择 $n=m=2$, $k=d=1$，则可以得到理想的辨识结果。在 MATLAB 工作空间中首先输

入表中的输入、输出数据，则调用辨识函数可以直接得出辨识结果。

```
>> r=[-1; -1; -1; -1; -1; -1; 1; -1; 1; -1; 1; -1; -1; 1; 1; -1;
      -1; 1; -1; -1;  -1; 1; -1; -1; 1; -1; 1; 1; -1; 1; 1; -1;
      -1; -1; 1; 1; 1; -1; 1; -1; -1; -1; -1; 1; 1; -1; 1; -1; 1;
       1; 1; -1; -1; 1; 1; 1; 1; -1; 1; 1; 1; 1; 1];
   y=[0; -1.0433; -1.0493; -0.7582; -0.9023; -0.9676; -0.7174;
      1.1856; -1.0971; 1.031; -0.9297; 0.8774; -0.7974; -1.2235;
      1.5932; 0.6827; -1.0792; -0.8278; 1.1727; -1.1212; -1.066;
      -0.6829; 0.8844; -0.8871; -0.9321; 1.1745; -1.0416; 1.1169;
      0.8535; -1.3582; 1.258; 0.8802; -1.1398; -0.811; -0.7764;
      1.2003; 0.9747; 0.886; -1.2276; 1.0991; -1.0116; -1.2223;
      -0.6089; -1.1249; 1.3905; 0.7324; -1.0738; 1.0923; -1.0962;
      0.7592; 1.1644; 0.5512; -0.8946; -0.9917; 1.4171; 0.8419;
      0.7732; 0.9675; -1.2756; 1.1331; 0.9793; 0.7447; 0.9256];
   U=iddata(y,r,0.05); H=armax(U,[2 2 1 1]); GG=tf(H);
   G1a=GG(1), G2a=GG(2)
   y1=lsim(G1a,r,t); plot(t,y,t,y1,':')
```

这时得出的 ARMX 辨识结果为

$$\frac{B(z)}{A(z)}=\frac{1.011z+0.8082}{z^2+0.9172z+0.1797},\quad \frac{C(z)}{A(z)}=\frac{0.09089z^2+0.04532z}{z^2+0.9172z+0.1797}$$

辨识系统的时域响应曲线与原始数据曲线如图 6-12 所示。可见，这样得出的辨识模型是比较准确的。

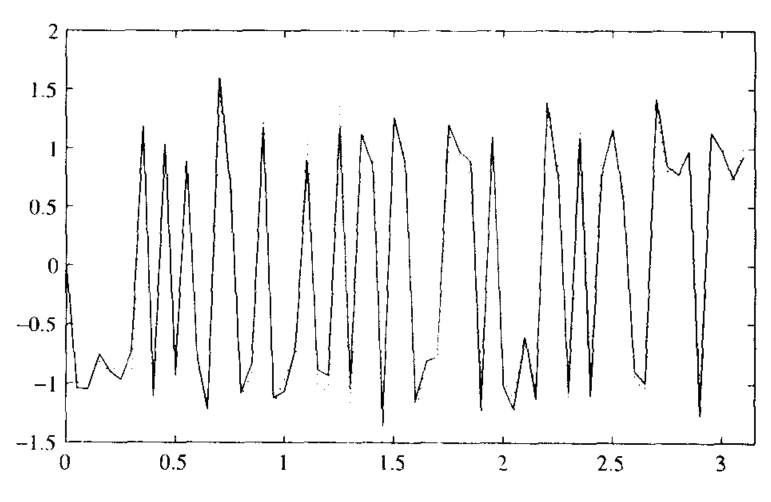


图 6-12 输出信号与辨识结果比较

6.4　自校正控制理论与仿真

自校正控制是自适应控制的一个重要的分支，其重要思想是与系统辨识算法相结合，逐步给出控制信号，达到对系统的有效控制。本节首先介绍在自校正控制领域重要的 Diophantine 方程的求解算法及其 MATLAB 实现，然后给出自适应预报的方法及仿真程序。本节还将介绍几个重要的自校正控制算法，如最小方差自校正调节器、控制器，极点配置的自校正调节器等。

6.4.1　Diophantine 方程及其求解

Diophantine 方程是自校正控制中很重要的方程，该方程是多项式方程

$$A(z^{-1})X(z^{-1})+B(z^{-1})Y(z^{-1})=C(z^{-1}) \tag{6-4-1}$$

其中 $A(z^{-1})=a_0+a_1z^{-1}+a_2z^{-2}+\cdots+a_nz^{-n}$，$B(z^{-1})=b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_mz^{-m}$，$C(z^{-1})=c_0+c_1z^{-1}+c_2z^{-2}+\cdots+c_kz^{-k}$，且 $m\leqslant n$。

该方程的解中，多项式 $X(z^{-1})$ 和 $Y(z^{-1})$ 的阶次应分别等于多项式 $B(z^{-1})$ 和 $A(z^{-1})$ 的阶次。下面将写出该方程的矩阵形式

$$\begin{bmatrix} a_0 & 0 & \cdots & 0 & b_0 & 0 & \cdots & 0\\ a_1 & a_0 & \ddots & 0 & b_1 & b_0 & \ddots & 0\\ a_2 & a_1 & \ddots & 0 & b_2 & b_1 & \ddots & 0\\ \vdots & \vdots & \ddots & a_0 & \vdots & \vdots & \ddots & b_0\\ a_n & a_{n-1} & \ddots & a_1 & b_n & b_{n-1} & \ddots & b_1\\ 0 & a_n & \ddots & a_2 & 0 & b_n & \ddots & b_2\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & a_n & 0 & 0 & \cdots & b_n \end{bmatrix}\begin{bmatrix} x_0\\ x_1\\ \vdots\\ x_m\\ y_0\\ y_1\\ \vdots\\ y_n\end{bmatrix}=\begin{bmatrix} c_1\\ c_2\\ \vdots\\ c_k\\ 0\\ \vdots\\ 0\end{bmatrix} \tag{6-4-2}$$

该方程左侧的系数矩阵称为 Sylvester 矩阵，其有惟一解的条件是 $A(z^{-1})$ 和 $B(z^{-1})$ 两个多项式互质。

从应用角度来说，该算法有一定的局限性。该算法只适用于 $k\leqslant n+m-1$ 的情况，如果 k 值过大，则原方程不存在惟一解。对于这种情况，应该考虑对 b_i 稍微扩展，从而得出满足方程的一个解。该方程的求解看起来较难，其实用 MATLAB 语言的几条语句就可以编写出求解的通用函数。

```
function [X,Y]=diopha_eq(A,B,C)
AAA=[]; A1=A(:); B1=B(:); n=length(B)-1; m=length(A)-1;
j=1; k=length(C); L=0; if k>n+m, L=k-m-n; end
```

```
for i=1:n, AAA(i:i+length(A1)-1,j)=A1; j=j+1; end
for i=1:m+L, AAA(i:i+length(B1)-1,j)=B1; j=j+1; end
C1=zeros(n+m,1); C1(1:length(C))=C(:); x=AAA\C1;
X=x(1:n)'; Y=x(n+1:end)';
```

例 6-13 已知某 Diophantine 方程中 $A(z^{-1})=0.212-1.249z^{-1}+2.75z^{-2}-2.7z^{-3}+z^{-4}$, $B(z^{-1})=2.04-1.2z^{-1}+3z^{-2}$, $C(z^{-1})=-0.36+0.6z^{-1}+2z^{-2}$, 试求解该方程。

求解 由下面的语句可以直接求出方程的解

```
>> A=[0.212,-1.249,2.75,-2.7,1]; B=[2.04,-1.2,3];
   C=[-0.36,0.6,2]; [X,Y]=diopha_eq(A,B,C)
```

调用该函数可以立即得出方程的解为 $X(z^{-1})=2.1289+0.9611z^{-1}$, $Y(z^{-1})=-0.3977+1.2637z^{-1}+0.0272z^{-2}-0.3204z^{-3}$。由 conv($A$,$X$)-conv($B$,$Y$) 语句则可见该结果与 C 向量完全一致。

例 6-14 考虑 Diophantine 方程，如果 $A(z^{-1})=1+2z^{-1}+3z^{-2}$, $B(z^{-1})=4+5z^{-1}+6z^{-2}$, $C(z^{-1})=2+4z^{-1}+6z^{-2}+8z^{-3}+10z^{-4}$, 试求解该方程。

求解 因为 $C(z^{-1})$ 的阶次过高，传统方法无法求解。这里采用 diopha_eq() 函数，用下面的语句直接求解方程，得出相应的解。

```
>> A=[1 2 3]; B=[4 5 6]; C=[2 4 6 8 10]; [X,Y]=diopha_eq(A,B,C)
   conv(A,X)+[conv(B,Y) 0]-C
```

由此可以得出方程的解为 $X(z^{-1})=-3.7778-4.5556z^{-1}$, $Y(z^{-1})=1.4444+2.2222z^{-1}+1.6667z^{-2}$，经检验满足原方程。值得指出的是，该解不是惟一的。可以类似地对多项式 $A(z^{-1})$ 增广，得出方程的另一个解。

6.4.2 提前 d 步预报算法与仿真

假设在第 t 时刻所有可以测出的输入输出数据为 $y(t),u(t),y(t-1),u(t-1),\cdots$，则由这些数据对 $t+d$ 时刻的输出进行预测，称为提前 d 步预测，记作 $\hat{y}(t+d\mid t)$。

使得预测误差的方差 $\mathrm{E}\left\{[y(t+d)-\hat{y}(t+d\mid t)]^2\right\}$ 为最小的提前 d 步预报信号满足下面的方程[7]

$$C(z^{-1})\hat{y}(t+d\mid t)=G(z^{-1})y(t)+F(z^{-1})u(t) \tag{6-4-3}$$

其中

$$F(z^{-1})=E(z^{-1})B(z^{-1}),\quad C(z^{-1})=A(z^{-1})E(z^{-1})+z^{-d}G(z^{-1}) \tag{6-4-4}$$

从上面的结论可见，可以先从后面的简化的 Diophantine 方程求出 $E(z^{-1})$ 和 $G(z^{-1})$，然后由前面的方程直接求出 $F(z^{-1})$。

例 6-15　已知某系统的离散模型为

$$y(t)-0.6y(t-1)+0.4y(t-2)=2u(t)+0.8\epsilon(t)+0.6\epsilon(t-1)+0.4\epsilon(t-2)$$

试求出提前两步的预测模型。

求解　由已知的方程可见，$A(z^{-1})=1-0.6z^{-1}+0.4z^{-2}$，$B(z^{-1})=2$，$C(z^{-1})=0.8+0.6z^{-1}+0.4z^{-2}$，且 $d=2$，这样可以由下面的语句输入这些多项式，并求出 $E(z^{-1})$，$F(z^{-1})$ 和 $G(z^{-1})$ 多项式。

```
>> A=[1 -0.6 0.6]; B=2; B1=[0 0 1]; C=[0.8 0.6 0.4];
   [E,G]=diopha_eq(A,B1,C), F=conv(E,B)
```

即 $E(z^{-1})=0.8+1.08z^{-1}$，$G(z^{-1})=0.5680-0.6480z^{-1}$，$F(z^{-1})=1.6+2.16z^{-1}$。这样可以将提前两步的预报方程写成

$$\hat{y}(t+2\,|\,t)=\frac{0.568-0.648z^{-1}}{0.8+0.6z^{-1}+0.4z^{-2}}y(t)+\frac{1.6+2.16z^{-1}}{0.8+0.6z^{-1}+0.4z^{-2}}u(t)$$

得出了预报方程，就可以建立起如图 6-13 所示的仿真模型，其中各个滤波模块在图中给出。假设信号发生器给出的是幅值为 4 的方波信号，采样周期 $T=0.01$ 秒，随机白噪声均值为 0，方差为 1，则可以得出预报信号与实际信号，如图 6-14 所示。

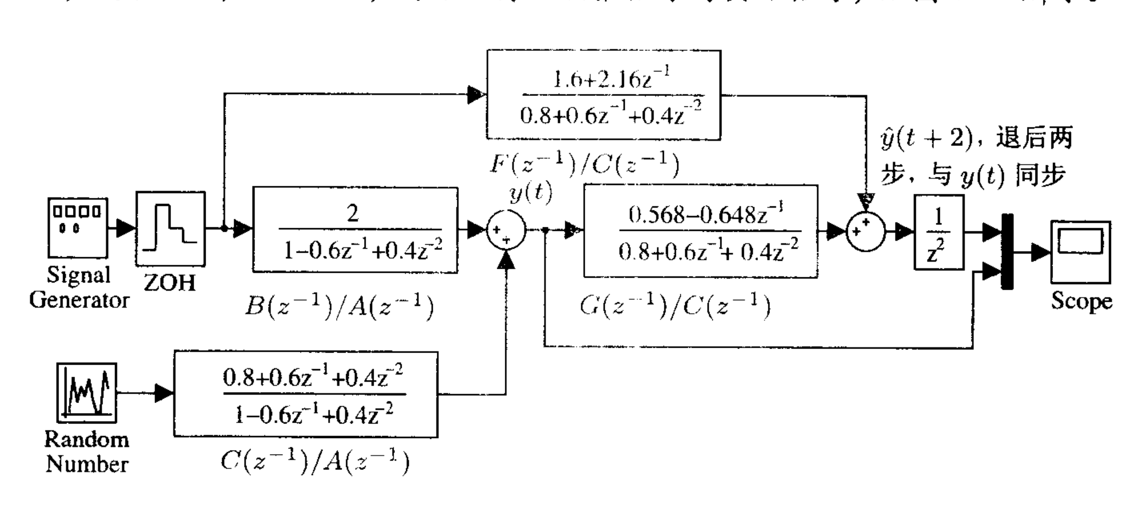


图 6-13　提前两步预报的仿真模型

6.4.3　最小方差自校正调节器与控制器

最小方差自校正调节器最早是由瑞典学者 Åström 和 Wittenmark 教授提出的[8]，其目标就是在存在噪声干扰的条件下，控制系统的输出值，使得其变化的方差最小。其基本思想是，在每一个采样周期内，用参数辨识的方法辨识出受控对象的模型系数，并根据这样的辨识参数计算出本采样周期内的控制量去控制一步系统模型。

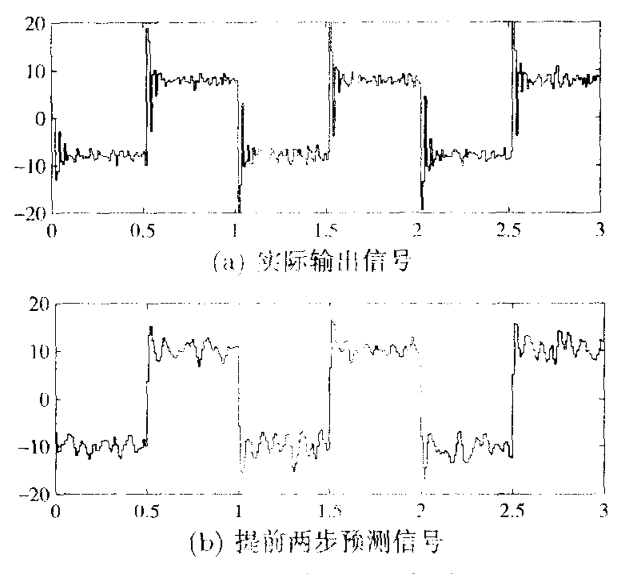


图 6-14 预报器仿真结果

受控对象的离散时间模型一般可以写为

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t) + C(z^{-1})\epsilon(t) \tag{6-4-5}$$

其中，纯延迟时间为 d，$\epsilon(t)$ 为零均值的白噪声信号。

1. 最小方差调节器

最小方差调节器设计的目标是得出控制序列 $\{u(t)\}$，使得实际输出 $y(t+d)$ 的方差为最小，即引入如下的目标函数

$$J = \min_{u} \mathrm{E}\left[y^2(t+d)\right] \tag{6-4-6}$$

如果受控对象的差分方程是给定的，即式 (6-4-5) 中的 $A(z^{-1})$, $B(z^{-1})$ 和 $C(z^{-1})$ 多项式已知，且延迟 d 给定，则使得指标式 (6-4-6) 最小的调节器控制律 $u(t)$ 可以由下面式子得出

$$u(t) = -\frac{G(z^{-1})}{B(z^{-1})E(z^{-1})}y(t) \tag{6-4-7}$$

式中，$G(z^{-1})$ 和 $E(z^{-1})$ 多项式满足下面的 Diophantine 方程

$$C(z^{-1}) = A(z^{-1})E(z^{-1}) + z^{-d}G(z^{-1}) \tag{6-4-8}$$

基于此算法，可以编写如下 MATLAB 程序来获得调节器增益传递函数。

```
function [H,E,G]=c6mstr1(A,B,C,d,T)
B1=[zeros(1,d) 1]; [E,G]=diopha_eq(A,B1,C);
H=tf(-[G,zeros(1,d-1)],conv(B,E),T); H.Variable='z^-1';
```

例 6-16 已知系统模型为[9]

$$y(t)-1.5y(t-1)+0.7y(t-2)=u(t-3)+0.5u(t-4)+\epsilon(t)+1.5\epsilon(t-1)+0.9\epsilon(t-2)$$

其中 $\epsilon(t)$ 为白噪声信号，系统采样周期 $T=0.1$。试求出其最小方差控制律。

求解 由系统模型可见，$d=3$，$A(z^{-1})=1-1.5z^{-1}+0.7z^{-2}$，$B(z^{-1})=1+1.5z^{-1}$，$C(z^{-1})=1+1.5z^{-1}+0.9z^{-2}$，这样由下面语句即可以求出最小方差控制律。

```
>> A=[1 -1.5 0.7]; B=[1 .5]; C=[1 1.5 0.9];
   [H E G]=c6mstr1(A,B,C,3,0.1)
```

得出的多项式 $E(z^{-1})=1+3z^{-1}+4.7z^{-2}$，且 $G(z^{-1})=4.95-3.29z^{-1}$，由此得出控制律为 $u(t)=\dfrac{-4.95+3.29z^{-1}}{1+3.5z^{-1}+6.2z^{-2}+2.35z^{-3}}y(t)$，或将控制律写成

$$u(t)=-3.5u(t-1)-6.2u(t-2)-2.35u(t-3)-4.95y(t)+3.29y(t-1)$$

由 Simulink 可以搭建出该最小方差控制系统的仿真模型，如图 6-15 (a) 所示。在该调节器的作用下，系统的输出曲线和控制信号分别如图 6-15 (b)、(c) 所示。

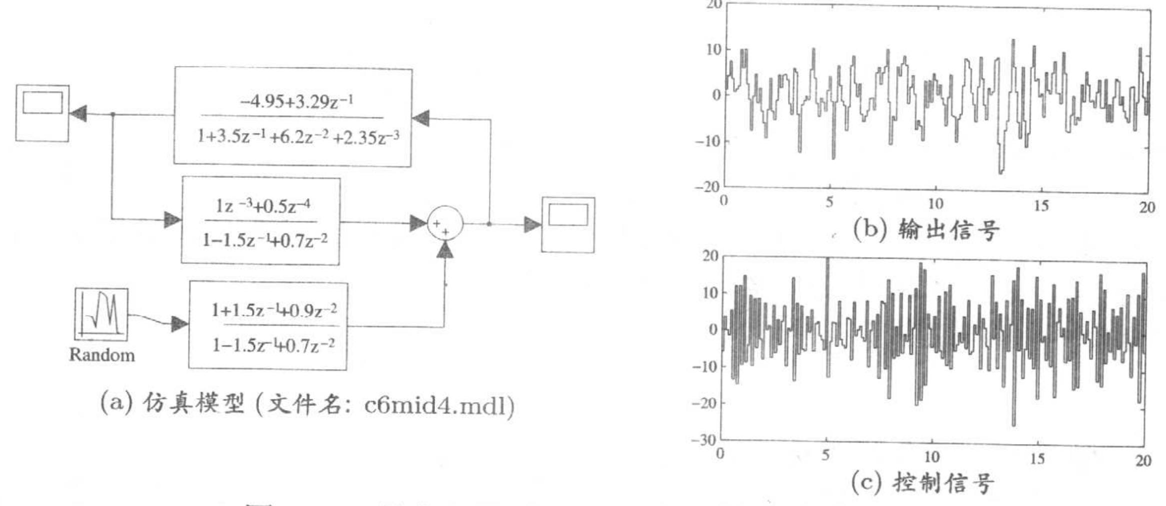


图 6-15 最小方差调节器仿真模型和控制效果

将得出的控制律式 (6-4-7) 代入模型式 (6-4-5)，结合 Diophantine 方程式 (6-4-8)，可以得出 $y(t)=E(z^{-1})\epsilon(t)$，亦即这时输出信号的方差最小。若多项式 $E(z^{-1})$ 可以表示为 $E(z^{-1})=e_0+e_1z^{-1}+\cdots+e_{d-1}z^{-d+1}$，且白噪声 $\epsilon(t)$ 的方差为 σ^2，则最小方差的值为 $\sum\limits_{i=0}^{d-1}e_i^2\sigma^2$。由上面的例子可以得出，输出信号 $y(t)$ 的最小方差为 $1+3^2+4.7^2=32.09$，其值较大。

从实质上看，最小方差调节器是用 $G(z^{-1})$ 的极点去对消受控对象的零点，所以这样的控制策略有致命的弱点，即它要求受控对象的零点是稳定的，亦即受控对象为最小相位系统。如果原系统不是最小相位的，则这里的控制律不能直接用于系统的控制。另外，如果受控对象的稳定裕度较小，则这样的调节器鲁棒性不

会太高。

2. 最小方差自校正调节器

如果受控对象参数是变化的或未知的，则可以将前面介绍的系统辨识方法结合起来，用递推辨识算法识别受控对象参数. 根据得出的参数计算控制信号，达到自适应调节的目的。

由前面的介绍显然可见，采用递推辨识的方法可以得出系统的 a_i, b_i 参数，然后通过求解相应的 Diophantine 方程则可以得出 $E(z^{-1})$ 和 $G(z^{-1})$ 多项式，再由式 (6-4-7) 计算出自适应律，达到自校正调节器设计的目的。这样的算法称为显式自校正算法。考虑这样的算法，因为每一步都需要求取 Diophantine 方程，使得算法实现较麻烦，所以可以考虑直接辨识自校正控制律的参数的方法，亦即所谓的隐式自校正算法。

假设需要调节的目标值为 $y_{\rm r}$。可以将提前 d 步的最优预报方程表示成

$$\begin{aligned} y(t+d|t) = {} & \alpha_1 y(t) + \cdots + \alpha_n y(t-n+1) \\ & + \beta_0 u(t) + \beta_1 u(t-1) + \cdots + \beta_{m+d-1} u(t-m-d+1) \end{aligned} \tag{6-4-9}$$

系统的估计方程可以写成

$$y(t) = \beta_0 u(t-d) + \boldsymbol{\psi}^{\rm T}(t-d)\boldsymbol{\theta}(t) + \epsilon(t) \tag{6-4-10}$$

其中从辨识结果的惟一性考虑，可以假设 β_0 为给定的值。但可以证明，该值选择是否“准确”不会影响整个自适应控制的效果[8]。另外，记

$$\boldsymbol{\psi}^{\rm T}(t) = \Big[y(t), y(t-1), \cdots, y(t-n+1), u(t-1), \cdots, u(t-m-d+1)\Big] \tag{6-4-11}$$

$$\boldsymbol{\theta}^{\rm T} = \Big[\alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_{m+d-1}\Big] \tag{6-4-12}$$

由前面介绍的递推辨识方法得

$$\boldsymbol{K}(t) = \frac{\boldsymbol{P}(t-1)\boldsymbol{\psi}(t-d)}{\lambda + \boldsymbol{\psi}^{\rm T}(t-d)\boldsymbol{P}(t-1)\boldsymbol{\psi}(t-d)} \tag{6-4-13}$$

$$\boldsymbol{P}(t) = \frac{1}{\lambda}\Big[\boldsymbol{P}(t-1) - \boldsymbol{K}(t)\boldsymbol{\psi}^{\rm T}(t-d)\boldsymbol{P}(t-1)\Big] \tag{6-4-14}$$

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}(t)\Big[y_{\rm r} - \beta_0 u(t-d) - \boldsymbol{\psi}^{\rm T}(t-d)\hat{\boldsymbol{\theta}}(t-1)\Big] \tag{6-4-15}$$

令式 (6-4-9) 左侧为 $y_{\rm r}$，则由辨识结果可以构造出自适应律

$$u(t) = \frac{1}{\beta_0}\Big[y_{\rm r} - \boldsymbol{\psi}^{\rm T}(t)\boldsymbol{\theta}(t)\Big] \tag{6-4-16}$$

仿照前面给出的递推最小二乘辨识模块及其 S-函数，可以建立起如图 6-16 所示的 Simulink 模块，其核心部分为由 S-函数编写的最小方差自校正调节器自适应律 $u(t)$ 的计算模块。注意，和系统辨识模块不同，算法同时涉及到 $\psi(t-d)$ 和 $\psi(t)$，所以这样的结构不适合高阶模型或大时间延迟模型的控制。在本模块中，有 12 路输入信号，分别为 $u(t-1),\cdots,u(t-6),y(t),\cdots,y(t-5)$，有 $m+n+d$ 路输出，其中第一路为计算出的自适应控制律 $u(t)$，其余的为辨识的参数。状态变量的选择类似于前面的系统辨识模块，因为辨识参数的个数发生了变化，辨识参数变成 $n+m+d-1$ 个，$\boldsymbol{P}$ 矩阵的维数可以相应地设置。这样，根据前面给出的隐式自校正调节算法，可以编写出下面的 S-函数。

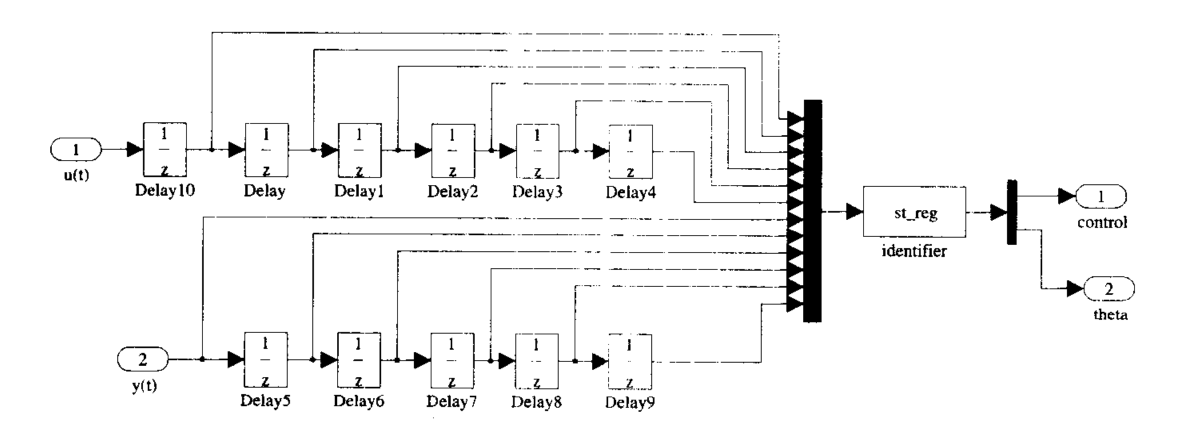


图 6-16　自校正调节器仿真模块

```
function [sys,x0,str,ts]=st_reg(t,x,u,flag,m,n,d,lam,b0,yr)
switch flag,
   case 0, [sys,x0,str,ts] = mdlInitializeSizes(m,n,d);
   case 2, sys=mdlUpdate(t,x,u,m,n,d,lam,b0);
   case 3, sys = mdlOutputs(t,x,u,m,n,d,b0,yr);
   case {1, 4, 9}, sys = [];
   otherwise % 错误处理
      error(['Unhandled flag = ',num2str(flag)]);
end;
%   初始化程序
function [sys,x0,str,ts] = mdlInitializeSizes(m,n,d)
s=simsizes; % 读入系统变量的默认值模板
s.NumContStates=0; s.NumDiscStates=(n+m+d)*(n+m+d-1);
s.NumOutputs=n+m+d; s.NumInputs=12; s.DirFeedthrough=1;
s.NumSampleTimes=1; sys=simsizes(s); % 设置系统模型变量
P0=1000*eye(n+m+d-1); x0=[zeros(n+m+d-1,1); P0(:)];
str=[]; ts=[-1 0]; % 继承输入信号的采样周期
%   离散状态更新函数
function sys = mdlUpdate(t,x,u,m,n,d,lam,b0)
psi_d=[u(7+d:6+d+n); u(1+d:m+d+1)]; theta=x(1:n+m+d-1);
```

```
PN=reshape(x(n+m+d:end),n+m+d-1,n+m+d-1);
K=PN*psi_d/(lam+psi_d'*PN*psi_d); PN=(PN-K*psi_d'*PN)/lam;
sys=[theta+K*(u(7)-b0*u(d)-psi_d'*theta); PN(:)];
%   输出计算函数
function sys = mdlOutputs(t,x,u,m,n,d,b0,yr)
psi=[u(7:6+n); u(1:m+1)]; theta=x(1:n+m+d-1);
sys=[(yr-psi'*theta)/b0; theta];
```

例 6-17 考虑一个二阶的受控对象模型[8]

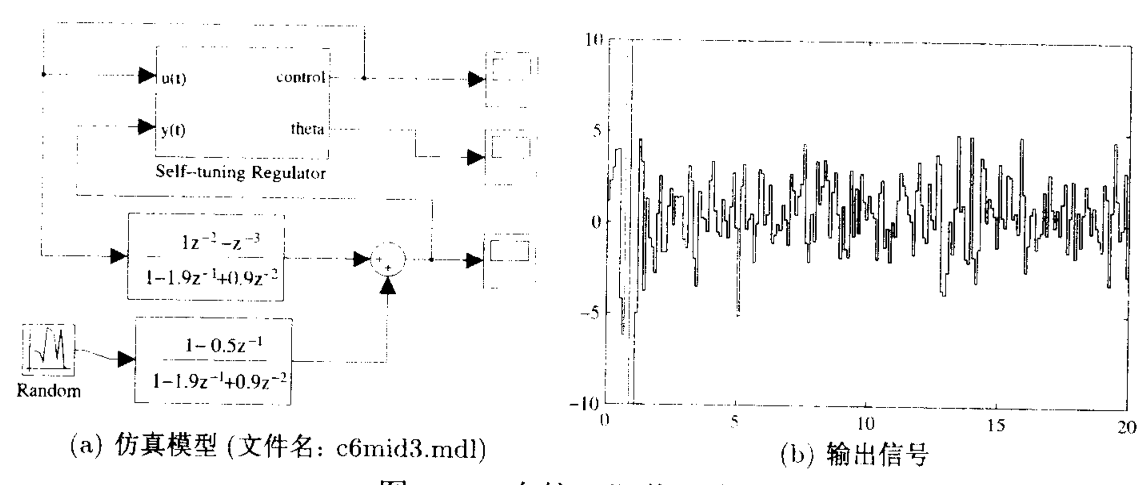
$$y(t)-1.9y(t-1)+0.9y(t-2)=u(t-2)-u(t-3)+\epsilon(t)-0.5\epsilon(t-1)$$

随机信号 $\epsilon(t)$ 为方差为 1 的 Gauss 过程，试得出 $y_r=10$ 的自校正控制效果。

求解 在前面的叙述中可见，自校正算法是基于 z^{-1} 的算子描述的，所以在 Simulink 下应该使用离散传递函数的滤波器表示模块。由原始模型，可以将受控对象改写成

$$y(t)=\frac{z^{-2}-z^{-3}}{1-1.9z^{-1}+0.9z^{-2}}u(t)+\frac{1-0.5z^{-1}}{1-1.9z^{-1}+0.9z^{-2}}\epsilon(t)$$

根据要求，可以搭建起如图 6-17 (a) 所示的系统仿真模型，其中，自校正调节器模块的相关参数可以设置为 $m=1$，$n=2$，$d=2$，$y_r=0$，$\beta_0=1$，$\lambda=1$。系统的采样周期为 $T=0.3$。在自校正调节器的作用下，输出信号如图 6-17 (b) 所示。



(a) 仿真模型 (文件名：c6mid3.mdl)　　(b) 输出信号

图 6-17 自校正调节器仿真

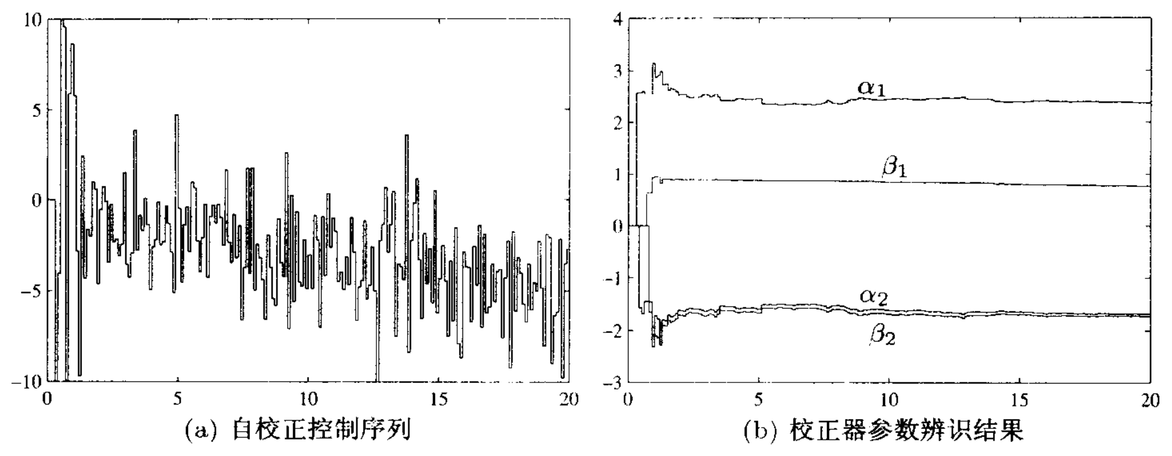
系统校正器的控制律和校正器辨识参数如图 6-18 所示。可见，为使得输出信号具有最小方差，则校正器产生的控制信号是很大的。

如果受控对象模型已知，则下面的语句可以得出最小方差校正器的数学模型。

```
>> A=[1 -1.9 0.9]; B=[1 -1]; C=[1 -0.5]; d=2;
   [H,E,G]=c6mstr1(A,B,C,d,0.1)
```

从而得出控制律为

$$H(z^{-1})=\frac{-1.76+1.26z^{-1}}{1+0.4z^{-1}-1.4z^{-2}}，即 u(t)=-1.76y(t)+1.26y(t-1)-0.4u(t-1)+1.4u(t-2)$$



(a) 自校正控制序列 (b) 校正器参数辨识结果

图 6-18 自校正调节器仿真结果

3. 最小方差自校正控制器

最小方差自校正控制器是由英国学者 Clarke 和 Gawthrop 教授提出的自适应控制方法[10]。和最小方差自校正调节器不同，控制器是为跟踪某理想信号设计的，而不是使得信号本身尽可能趋于零。最小方差自适应控制器设计的目标是得出控制序列 $u(t)$，使得实际输出 $y(t+d)$ 与期望输出 $y_{\rm r}(t+d)$ 之间的方差为最小，即引入如下的目标函数

$$J=\min_{u}\ {\rm E}\left\{\left[y(t+d)-y_{\rm r}(t+d)\right]^2\right\} \tag{6-4-17}$$

4. 加权最小方差控制器

前面介绍的自校正控制策略均未考虑输入信号的大小，而在实际控制中应该对输入信号有两种考虑，其一，控制信号的大小将反映出控制行为所需的能量，另一方面，有的系统不允许大的控制信号，因为大信号可能导致系统硬件出现问题，所以在实际控制中，应该将控制信号的大小考虑在内。另外，前面介绍的最小方差自校正调节器不适用于非最小相位受控对象的控制与调节。

由于式 (6-4-17) 中给出的目标函数中未考虑输入量，所以可以引入新的目标函数

$$J=\min_{u}\ {\rm E}\left\{\left[P(z^{-1})y(t+d)-R(z^{-1})r(t)\right]^2+\left[\varLambda(z^{-1})u(t)\right]^2\right\} \tag{6-4-18}$$

其中 $r(t)$ 为参考输出信号，多项式 $P(z^{-1})=1+p_1z^{-1}+\cdots+p_{n_p}z^{-n_p}$，$R(z^{-1})=r_0+r_1z^{-1}+\cdots+r_{n_r}z^{-n_r}$，$\varLambda(z^{-1})=\lambda_0+\lambda_1z^{-1}+\cdots+\lambda_{n_\lambda}z^{-n_\lambda}$ 分别为对实际输出信号、期望输出信号与控制信号的加权函数。为得出该问题的最优解，可以引入一个辅助系统

$$S(t+d)=\psi(t+d)-R(z^{-1})r(t)+\frac{\lambda_0}{b_0}\varLambda(z^{-1})u(t) \tag{6-4-19}$$

式中 $\psi(t+d)=P(z^{-})y(t+d)$。这样，广义最小方差自校正控制问题就可以转换成前面介绍的最小方差自适应调节问题，写出其自适应律为

$$u(t)=\frac{R(z^{-1})r(t)-\psi(t+d|t)}{\dfrac{\lambda_0}{b_0}\Lambda(z^{-1})} \tag{6-4-20}$$

考虑式 (6-4-3) 中给出的自适应预报律，其变化形式为

$$\hat{y}(t+d\mid t)=\frac{G(z^{-1})}{C(z^{-1})}y(t)+\frac{F(z^{-1})}{C(z^{-1})}u(t) \tag{6-4-21}$$

将其代入式 (6-4-20) 则可以得出广义最小方差控制律为

$$u(t)=\frac{C(z^{-1})R(z^{-1})r(t)-G(z^{-1})y(t)}{\dfrac{\lambda_0}{b_0}C(z^{-1})\Lambda(z^{-1})+B(z^{-1})E(z^{-1})} \tag{6-4-22}$$

根据上面给出的控制律算法，可以编写出下面的 MATLAB 函数

```
function [den,num1,num2]=st_contr(A,B,C,d,P,R,Lam)
B1=[zeros(1,d) 1]; [E,G]=diopha_eq(A,B1,C);
lam0=Lam(1); b0=B(1); den=lam0/b0*conv(Lam,C)+conv(B,E);
num2=-G; num1=conv(C,R);
```

例 6-18 考虑一个简单的广义最小方差控制问题，假设系统的模型为[7]

$$y(t)=1.5y(t-1)-0.7y(t-2)+u(t-1)+0.5u(t-2)+\epsilon(t)-0.5\epsilon(t-1)$$

目标函数退化为 $J=\mathrm{E}\left\{[y(t+1)-r(t)]^2+0.5u^2(t)\right\}$，试求控制律模型。

求解 由给出的方程式可见，$A(z^{-1})=1-1.5z^{-1}+0.7z^{-2}$，$B(z^{-1})=1+0.5z^{-1}$，$C(z^{-1})=1-0.5z^{-1}$，$P(z^{-1})=R(z^{-1})=1$，$\Lambda(z^{-1})=\sqrt{0.5}$，$d=1$。可以先求出多项式 $E(z^{-1}),F(z^{-1}),G(z^{-1})$，进而求出广义最小方差控制律。

```
>> A=[1 -1.5 0.7]; B=[1 0.5]; C=[1 -0.5]; d=1; P=1; R=1;
   Lam=sqrt(0.5); [den,n1,n2]=st_contr(A,B,C,d,P,R,Lam)
```

由得出的三个多项式可以写出控制律为

$$u(t)=\frac{(1-0.5z^{-1})r(t)-(1-0.7z^{-1})y(t)}{1.5+0.25z^{-1}}$$

控制律还可以写成

$$u(t)=\frac{1}{1.5}\Big[r(t)-0.5r(t-1)-y(t)+0.7y(t-1)-0.25u(t-1)\Big]$$

仿照最小方差自校正调节器的算法与 S-函数实现，可以同样编写出广义最小方差自校正控制器设计的 S-函数。

6.4.4 极点配置控制器设计

和连续状态方程模型的极点配置类似，离散系统极点配置的目标也是通过某些算法将闭环系统的极点配置到某些预先指定的位置上，通过这样的方法来改善离散系统的性能。

假设受控对象模型的数学表示为

$$A(z^{-1})y(t) = B(z^{-1})u(t) + v(t) \tag{6-4-23}$$

其中 $v(t)$ 为扰动信号，$A(z^{-1})$ 为首一多项式。

假设控制器可以写成

$$R(z^{-1})u(t) = T(z^{-1})r(t) - S(z^{-1})y(t) \tag{6-4-24}$$

将控制信号代入式 (6-4-23)，则可以得出闭环模型

$$y(t) = \frac{B(z^{-1})T(z^{-1})}{A(z^{-1})R(z^{-1})+B(z^{-1})S(z^{-1})}r(t) + \frac{R(z^{-1})}{A(z^{-1})R(z^{-1})+B(z^{-1})S(z^{-1})}v(t) \tag{6-4-25}$$

再假设从参考输入信号 $r(t)$ 驱动，期望得到的输出信号 $y_{\rm m}(t)$ 可以由下面的参考模型得出

$$A_{\rm m}(z^{-1})y_{\rm m}(t) = B_{\rm m}(z^{-1})r(t) \tag{6-4-26}$$

则可以令二者相等，得出

$$\frac{B(z^{-1})T(z^{-1})}{A(z^{-1})R(z^{-1}) + B(z^{-1})S(z^{-1})} = \frac{B_{\rm m}(z^{-1})}{A_{\rm m}(z^{-1})} \tag{6-4-27}$$

注意，这里可能设计零极点对消。所以为保证闭环系统的稳定性，应该确保只有稳定的零极点发生对消，否则闭环系统在实际使用时将出现不稳定现象。可以将 $B(z^{-1})$ 多项式分解成稳定的部分 $B^+(z^{-1})$ 和不稳定的部分 $B^-(z^{-1})$，即记 $B(z^{-1}) = B^+(z^{-1})B^-(z^{-1})$。假设 $B^+(z^{-1})$ 多项式的零点全部被对消，则闭环系统极点除了 $A_{\rm m}(z^{-1})$ 还应考虑观测器多项式 $A_{\rm o}(z^{-1})$，这样，根据 $B(z^{-1})$ 多项式零极点对消情况应该考虑三种情况。

① **无零极点对消**：$B(z^{-1}) = B^-(z^{-1})$，$B^+(z^{-1}) = 1$，则通过求解下面的 Diophantine 方程

$$A(z^{-1})R(z^{-1}) + B(z^{-1})S(z^{-1}) = A_0(z^{-1})A_{\rm m}(z^{-1}) \tag{6-4-28}$$

可以得出多项式 $R(z^{-1})$ 和 $S(z^{-1})$，并由此求出多项式 $T(z^{-1})$，并构造出自适应控制律

$$T(z^{-1}) = A_0(z^{-1})\frac{B_{\rm m}(z^{-1})}{B(z^{-1})} \tag{6-4-29}$$

② **全部零点对消**：即 $B^+(z^{-1})$ 包含 $B(z^{-1})$ 的全部零点。这时，$B(z^{-1}) = b_0B^+(z^{-1})$，可以求出下面 Diophantine 方程的解 $R_1(z^{-1})$，$S(z^{-1})$

$$A(z^{-1})R_1(z^{-1}) + b_0S(z^{-1}) = A_0(z^{-1})A_{\rm m}(z^{-1}) \tag{6-4-30}$$

从而得出自适应控制律

$$R(z^{-1}) = R_1(z^{-1})B^+(z^{-1}),\ \ T(z^{-1}) = \frac{1}{b_0}A_0(z^{-1})B_{\rm m}(z^{-1}) \tag{6-4-31}$$

③ **部分零极点对消**：通过求解 Diophantine 方程

$$A(z^{-1})R^-(z^{-1}) + B^-(z^{-1})S(z^{-1}) = A_0(z^{-1})A_{\rm m}(z^{-1}) \tag{6-4-32}$$

可以得出 $R_1(z^{-1})$ 和 $S(z^{-1})$，并根据下面的式子得出自适应律

$$R(z^{-1}) = R_1(z^{-1})B^+(z^{-1}),\ \ T(z^{-1}) = A_0(z^{-1})\frac{B_{\rm m}(z^{-1})}{B^-(z^{-1})} \tag{6-4-33}$$

得出了相关的多项式，则可以得出自适应律为

$$T_{\rm d}(z^{-1})R(z^{-1})u(t) = T_{\rm n}(z^{-1})r(t) - T_{\rm d}(z^{-1})S(z^{-1})y(t) \tag{6-4-34}$$

根据前面的算法，可以编写出如下的 MATLAB 函数来求取自适应律

```
function [R,S,Tn,Td]=pp_str(A,B,d,A0,Am,Bm)
p=roots(B); i1=find(abs(p)>1); i2=find(abs(p)<=1);
Tn=conv(A0,Bm); Bz=zeros(1,d);
Td=Bm; B1=poly(p(i1)); B2=poly(p(i2)); b0=B(1); B2=b0*B2;
if length(i1)==0
   [R,S]=diopha_eq(A,[Bz B],conv(A0,Am));
elseif length(i2)==0
   [R1,S]=diopha_eq(A,[Bz b0],conv(A0,Am)); R=conv(R1,B1); Td=b0;
else
  [R1,S]=diopha_eq(A,[Bz B2],conv(A0,Am)); R=conv(R1,B1); Td=B2;
end
```

例 6-19 考虑连续系统 $G(s) = \dfrac{1}{s(s+1)}$，若参考模型为 $G_{\rm m}(s) = \dfrac{\omega_{\rm n}^2}{s^2 + 2\zeta\omega_{\rm n}s + \omega_{\rm n}^2}$，且 $\zeta = 0.707, \omega_{\rm n} = 1$，采样周期 $T = 0.5$。选择观测器 $A_0 = (z-0.5)^2$，试求出极点配置的控制器模型[6]。

求解 输入连续的受控对象模型和参考模型，对其离散化，则可以得出 $A, B, A_{\rm m}, B_{\rm m}$

```
>> s=tf('s'); G=1/s/(s+1); Gm=tf(1,[1 2*0.707 1]);
   G1=c2d(G,0.5), Gm1=c2d(Gm,0.5)
```

这样可以得出

$$G_1 = \frac{0.1065z + 0.0902}{z^2 - 1.607z + 0.6065},\ G_{\mathrm{m}_1}(z) = \frac{0.09812z + 0.07744}{z^2 - 1.318z + 0.4931}$$

若将离散模型写成 z^{-1} 的形式，可以看出 $d = 1$。

```
>> B=[0.1065,0.0902]; A=[1,-1.607,0.6065]; d=1;
   Bm=[0.09812,0.07744]; Am=[1,-1.318,0.4931]; A0=[1,-1,0.25];
   [R,S,Tn,Td]=pp_str(A,B,d,A0,[1 0 0],Am,Bm), conv(Td,R),
```

由上述语句可以直接得出相关的各个多项式为

$R(z^{-1}) = 1.0000 - 1.0094z^{-1}$, $S(z^{-1}) = 2.8020 - 3.9462z^{-1} + 1.3667z^{-2}$

$T_{\mathrm{n}}(z^{-1}) = 0.0981 - 0.0207z^{-1} - 0.0529z^{-2} + 0.0194z^{-3}$

$T_{\mathrm{d}}(z^{-1})R(z^{-1}) = 0.0981 - 0.0216z^{-1} - 0.0782z^{-2}$

这样，控制律可以写成

$$u(t) = \frac{0.098-0.021z^{-1}-0.053z^{-2}+0.0194z^{-3}}{0.0981-0.0216z^{-1}-0.0782z^{-2}}r(t) - \frac{2.802-3.946z^{-1}+1.367z^{-2}}{1.0000-1.0094z^{-1}}y(t)$$

6.5 预测控制系统及仿真

预测控制是基于模型的预测算法[11]，具体的模型可以采用对象的历史数据，对线性系统来说也可以是状态方程、传递函数模型。对稳定的线性系统来说，这里的模型还可以采用系统的阶跃响应或脉冲效应数据作为预测的模型，由这些模型预测系统的输出信号，将其和实际输出信号比较，通过优化的方法计算出一系列输入信号 $u(t), \cdots, u(t_N)$，由 $u(t)$ 信号控制一步。采用这样递推的控制方法的策略又称为模型预测控制。模型预测控制系统的框图如图 6-19 所示。

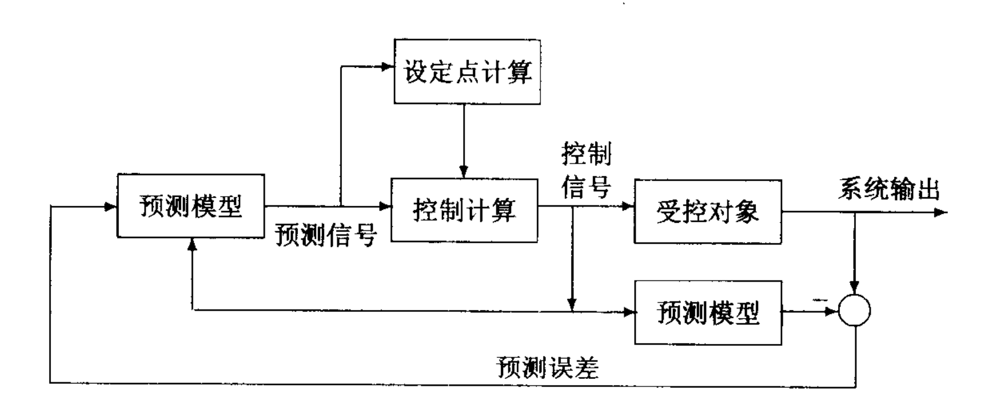


图 6-19 模型预测控制框图

参考图 6-20 中给出的输出、控制信号示意图可见，预测控制的基本想法是计

算控制量序列 $u(t)$，使得系统的预期输出信号和根据当前输入输出数据之间的某种性能指标为最小。

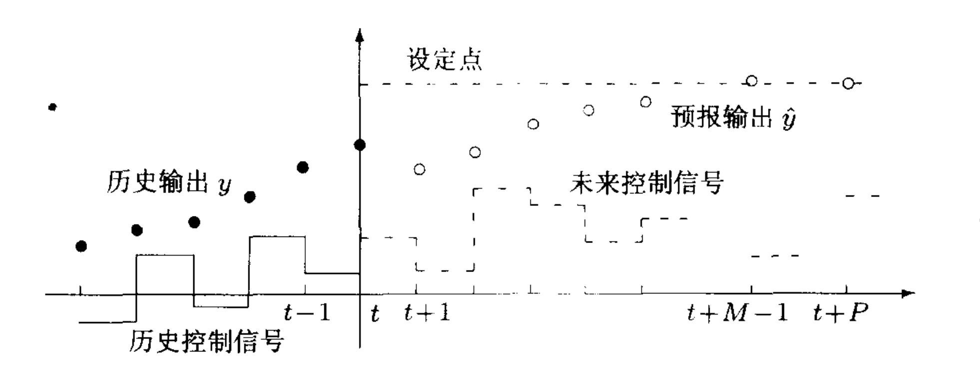


图 6-20 输入、控制信号示意图

6.5.1 动态矩阵控制方法

假设已知某线性系统的在一些采样点处的阶跃响应值为 $s_1, s_2, \cdots, s_P$，如图 6-21 所示。这时，系统在任意时刻 t 的阶跃响应可以写成

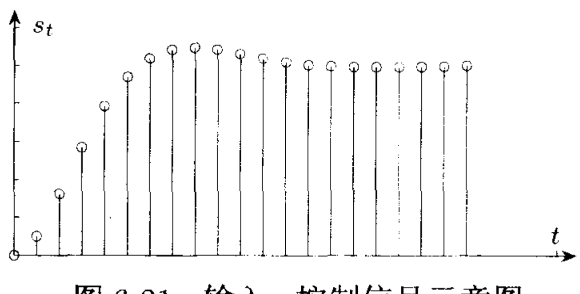


图 6-21 输入、控制信号示意图

$$y(t) = \sum_{i=1}^{P} s_i \Delta u(t-i) = \sum_{i=1}^{P-1} s_i \Delta u(t-i) + s_P \Delta u(t-P) \tag{6-5-1}$$

其中 $\Delta u(t)$ 是在 t 时刻的控制增量。注意，P 的选择应该使得阶跃响应进入稳态区域，否则这样的表述没有实际意义。为方便叙述动态矩阵控制方法起见，单独将 $u(t-P)$ 项分离出来。利用上式可以容易地写出输出信号的前 n 步预报估计

$$\hat{y}(t+j) = \sum_{i=1}^{P-1} s_i \Delta u(t+j-i) + s_P \Delta u(t+j-P) \tag{6-5-2}$$

其中，$j = 1, 2, \cdots, n$。从这里给出的输入信号看，对照图 6-21 可见，这里的输入信号部分是过去的信号，即下标 $j-i<0$ 的信号，其余的为当前和将来的信号，

所以有必要将它们从上式中分离出来，这样

$$\hat{y}(t+j)=\sum_{i=1}^{j}s_i\Delta u(t+j-i)+\sum_{i=j+1}^{P-1}s_i\Delta u(t+j-i)+s_P\Delta u(t+j-P) \quad (6\text{-}5\text{-}3)$$

记上式后两项之和为 $\tilde{y}(t+j)$

$$\tilde{y}(t+j)=\sum_{i=j+1}^{P-1}s_i\Delta u(t+j-i)+s_P\Delta u(t+j-P) \quad (6\text{-}5\text{-}4)$$

可见，$\tilde{y}(t+j)$ 为根据过去已经发生的信息对输出的估计。更简单地，上面各个预估信号可以表示为矩阵形式

$$\begin{bmatrix}\hat{y}(t+1)\\ \hat{y}(t+2)\\ \vdots\\ \hat{y}(t+n)\end{bmatrix}=\begin{bmatrix}s_1 & & & \\ s_2 & s_1 & & \\ \vdots & \vdots & \ddots & \\ s_n & s_{n-1} & \cdots & s_1\end{bmatrix}\begin{bmatrix}\Delta u(t)\\ \Delta u(t+1)\\ \vdots\\ \Delta u(t-P)\end{bmatrix}+\begin{bmatrix}\tilde{y}(t+1)\\ \tilde{y}(t+2)\\ \vdots\\ \tilde{y}(t+n)\end{bmatrix} \quad (6\text{-}5\text{-}5)$$

如果不采用全部的 n 项输入，只部分地选择其中 M 项，则可能在不牺牲性能的前提下，减小计算量。这时上述的模型可以改写成

$$\begin{bmatrix}\hat{y}(t+1)\\ \hat{y}(t+2)\\ \vdots\\ \hat{y}(t+n)\end{bmatrix}=\begin{bmatrix}s_1 & & & \\ s_2 & s_1 & & \\ \vdots & \vdots & \ddots & \\ s_n & s_{n-1} & \cdots & s_{n+1-M}\end{bmatrix}\begin{bmatrix}\Delta u(t)\\ \Delta u(t+1)\\ \vdots\\ \Delta u(t-M)\end{bmatrix}+\begin{bmatrix}\tilde{y}(t+1)\\ \tilde{y}(t+2)\\ \vdots\\ \tilde{y}(t+n)\end{bmatrix} \quad (6\text{-}5\text{-}6)$$

该方程可以用矩阵形式表述成

$$\hat{\boldsymbol{y}}(t+1)=\boldsymbol{S}\Delta\boldsymbol{u}(t)+\tilde{\boldsymbol{y}}(t+1) \quad (6\text{-}5\text{-}7)$$

其中 $\hat{\boldsymbol{y}}(t+1)$, $\boldsymbol{S}$, $\Delta\boldsymbol{u}(t)$, $\tilde{\boldsymbol{y}}(t+1)$ 的定义由上式显而易见。这里的 $\boldsymbol{S}$ 矩阵称为动态矩阵，因为该矩阵的各个元素是描述系统动态参数的，从另一个角度看，$\boldsymbol{S}$ 是一个常数矩阵。如果已知的受控对象为非线性的，也可以求出工作点附近阶跃响应的实际构成。引入最优化的目标函数

$$J=\sum_{j=1}^{P}q(j)\Big[\hat{y}(t+j)-w(t+j)\Big]^2+\sum_{j=1}^{M}r(j)\Delta u^2(t+j-1) \quad (6\text{-}5\text{-}8)$$

其中 $w(t+j)$ 为期望的输出值，$\lambda(j)$ 为加权量。P,M 分别称为预测时域和控制时域。对当前时刻可以通过求解最优化问题，得出控制序列 $u(t)$。由得出的序列

对原系统控制一步，则需要再求解最优化问题更新控制序列，所以这样的最优化问题又属于滚动时域 (receding horizon) 控制问题[11]。

对照前面介绍的最优化方法，如果 $u(t)$ 本身可以取任何的值则原问题即为无约束最优化问题，否则即为有约束的最优化问题，采用数值方法即可以直接求解。MATLAB 的预测控制工具箱[12]提供了相应函数可以求解这样的问题。新版本的工具箱主要侧重于直接的界面求解，本节先介绍基于函数调用的设计方法，然后给出基于界面的设计方法，并介绍基于 Simulink 的仿真方法。

1. 系统模型与阶跃响应模型描述

如果受控对象为线性系统，则可以通过 `poly2tfd()` 函数描述其传递函数模型，其调用格式为 G=`poly2tfd(`$\boldsymbol{n}$,$\boldsymbol{d}$,T,τ`)`，其中 $\boldsymbol{n},\boldsymbol{d}$ 为传递函数分子和分母多项式系数，T 为采样周期，τ 为延迟时间常数。若给出的是连续系统模型，则应该将 T 设置为 0。这里的 G 为单变量系统模型，若想处理多变量模型，则应该有一系列这样的单变量模型来描述。

由定义的传递函数模型，可以通过 `mod=tfd2step(`$t_{\rm f}$,T,k,G`)` 函数建立阶跃响应模型 `mod`，其中 $t_{\rm f}$ 为终止时间，k 为输出信号的标志位，若进入稳态则赋 1，否则为 0。如果已知多变量模型 $g_{11},g_{12},\cdots,g_{nm}$，则该函数的调用格式为 `mod=tfd2step(`$t_{\rm f}$,$T$,$\boldsymbol{k}$,$g_{11},g_{12},\cdots,g_{mm}$`)`，其中 $\boldsymbol{k}$ 为对应每路输出信号的标志位。`plotstep(mod)` 函数可以绘制模型的阶跃响应曲线。

当然，系统的阶跃响应模型可以通过 `step()` 类函数构造，具体的系统构造格式可以参见该工具箱手册。

例 6-20 假设某受控对象模型为 $G(s)=\dfrac{\mathrm{e}^{-s}}{(s+1)(2s+1)}$，采样周期为 $T=0.5$，试表示其阶跃响应模型。

求解 对稳定系统来说，应该选择一个合适的终止仿真时间使输出信号进入稳态。通过试凑可见，若选择终止仿真时间为 $t_{\rm f}=12$ 则可以进入稳态，这样可以通过下面的命令得出该系统的阶跃响应模型。注意，这时得出的提示为，最后一点的阶跃响应值的稳态误差是 0.82%，说明已经进入稳态了。系统阶跃响应数据如图 6-22 所示。

```
>> t_f=12; T=0.5; G=poly2tfd(1,conv([1 1],[2 1]),0,1);
   mod=tfd2step(t_f,T,1,G); t=0:T:t_f;
   y=mod(1:length(t)); stem(t,y)
```

例 6-21 考虑下面的多变量系统模型，试建立其阶跃响应模型。

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.1134\mathrm{e}^{-0.72s}}{1.78s^2+4.48s+1} & \dfrac{0.924}{2.07s+1}\\[2mm] \dfrac{0.3378\mathrm{e}^{-0.3s}}{0.361s^2+1.09s+1} & \dfrac{-0.318\mathrm{e}^{-1.29s}}{2.93s+1}\end{bmatrix}$$

求解 可以给出下面语句输入该多变量系统的数学模型，并求出其阶跃响应，绘制出

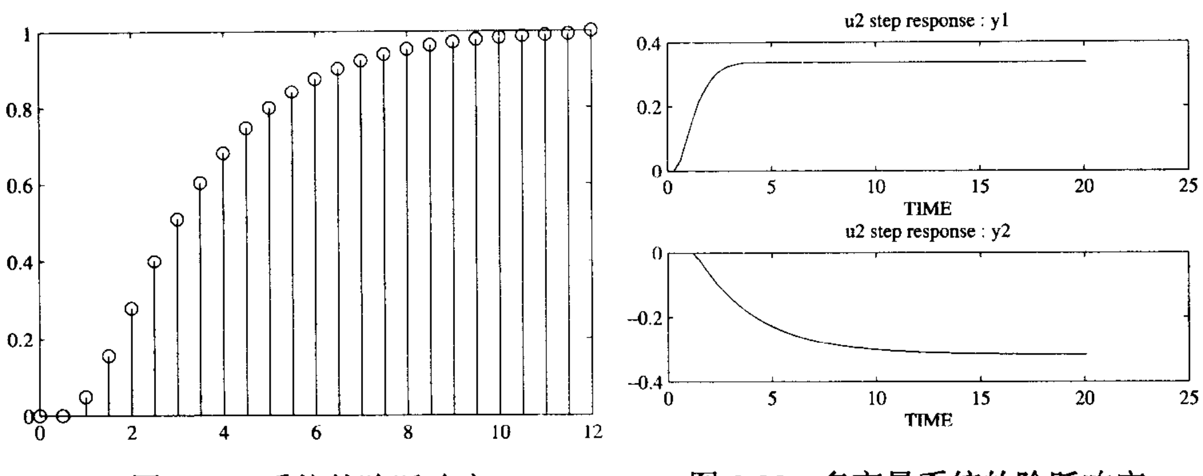


图 6-22 系统的阶跃响应 图 6-23 多变量系统的阶跃响应

阶跃响应曲线，如图 6-23 所示。

```
>> g11=poly2tfd(0.1134,[1.78,4.48,1],0,0.72);
   g12=poly2tfd(0.924,[2.07,1],0,0);
   g21=poly2tfd(0.3378,[0.361,1.09,1],0,0.3);
   g22=poly2tfd(-0.318,[2.93,1],0,1.29);
   S=tfd2step(20,0.3,[1 1],g11,g12,g21,g22); plotstep(S)
```

2. 无约束最优预测控制设计

无约束的最优预测控制器可以调用 mpccon() 函数直接设计

$\boldsymbol{K}_{\rm mpc}$=mpccon(mod,$\boldsymbol{q}$,$\boldsymbol{r}$,M,P)

其中 mod 为由阶跃响应数据构成的变量，$\boldsymbol{q}$, $\boldsymbol{r}$ 向量分别表示对误差与输出增量的加权，M 和 P 分别为控制时域与预测时域。得出的 $\boldsymbol{K}_{\rm mpc}$ 为控制器的增益矩阵。得出控制增益后，则可以由 mpcsim() 函数得出预测控制下系统的响应曲线

[$\boldsymbol{y}$,$\boldsymbol{u}$,$\boldsymbol{y}_{\rm m}$]=mpcsim(plant,mod,$\boldsymbol{K}_{\rm mpc}$,$t_{\rm end}$,$s_{\rm p}$)

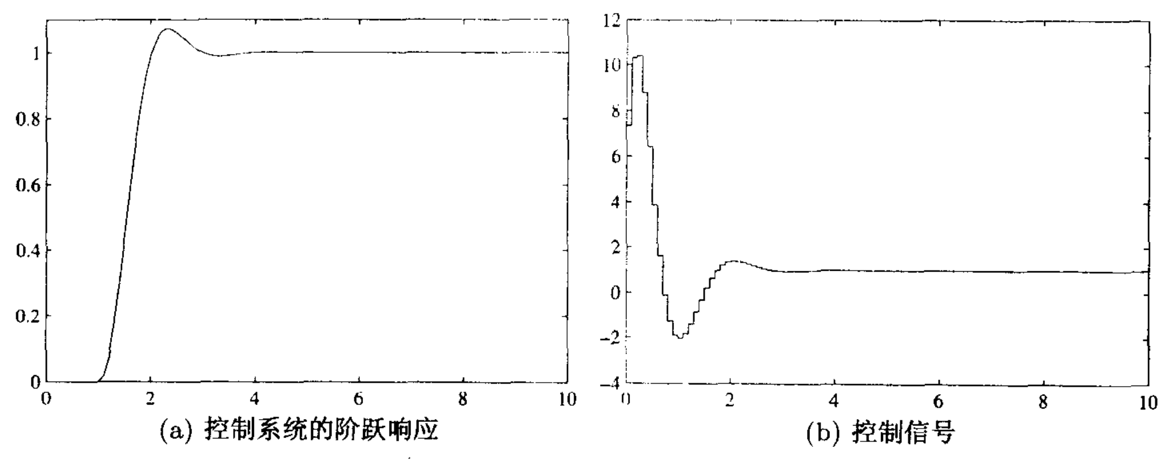
其中，plant 为受控对象模型的阶跃响应数据，mod 为设计控制器时使用的阶跃响应数据，二者可以相同也可以不同。$t_{\rm end}$ 为终止仿真时间，$s_{\rm p}$ 为参考输入信号。得到的 $\boldsymbol{y}$ 为系统的输出，$\boldsymbol{u}$ 为控制序列，$\boldsymbol{y}_{\rm m}$ 为模型输出。

例 6-22 考虑例 6-20 的受控对象模型，设计预测控制器并通过仿真分析系统的响应。

求解 采用加权 $q1,r = 0.1$，令预测时域 $P = 40$，选择控制时域 $M = 10$，由下面的语句可以设计出模型预测控制器，并绘制出系统的输出曲线和控制信号，如图 6-24 (a)、(b) 所示。

```
>> t_f=12; T=0.1; G=poly2tfd(1,conv([1 1],[2 1]),0,1);
   mod=tfd2step(t_f,T,1,G); P=40;M=10; q=1; r=0.1;
   Kmpc=mpccon(mod,q,r,M,P); plant=mod; sp=1;
```

```
tend=10; t=0:T:tend; [y,u]=mpcsim(plant,mod,Kmpc,tend,sp);
plot(t,y), figure, stairs(t,u)
```



(a) 控制系统的阶跃响应　　(b) 控制信号

图 6-24　预测控制器的仿真结果

从控制结果可见，输出效果非常好，但这种效果的代价是极大的输入信号，这有时在实际应用中是没有实际意义的。所以对这种系统的控制应该考虑采用有约束的最优化方法来实现。

利用上面的同样语句，还可以绘制出不同的 m 取值下的控制效果，如图 6-25 所示。如果选择大些的 m，如 $m = 15$，输出曲线和 $m = 10$ 时相仿，若减小 m 的值，如令 $m = 5$，则曲线有很大不同。这表示控制时域 m 对控制效果有显著影响，如果时域选择不足，得出的控制效果不能完全达到模型预测控制的目标，MATLAB 程序如下：

```
>> K1=mpccon(mod,q,r,10,P); [y1,u]=mpcsim(plant,mod,K1,tend,sp);
   K2=mpccon(mod,q,r,5,P);  [y2,u]=mpcsim(plant,mod,K2,tend,sp);
   K3=mpccon(mod,q,r,20,P); [y3,u]=mpcsim(plant,mod,K3,tend,sp);
   plot(t,y1,t,y2,'-',t,y3,':')
```

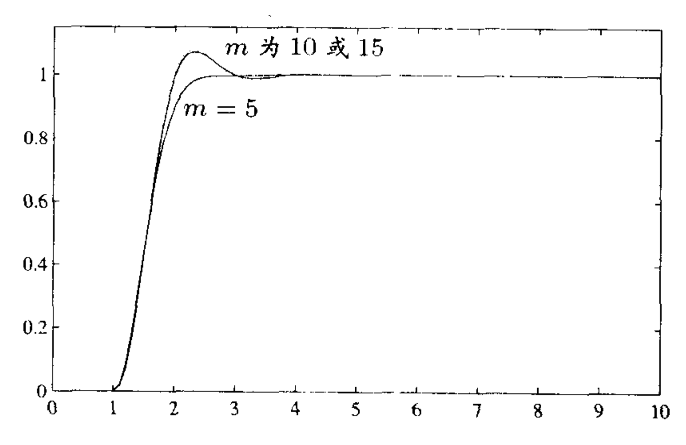


图 6-25　不同控制时域对控制效果的影响

3. 有约束最优预测控制设计

如果对系统的控制信号 $\boldsymbol{u}$ 和输出信号引入约束，则前面介绍的预测控制演化成有约束最优化问题的求解。这样的问题可以由 cmpc() 函数直接求解

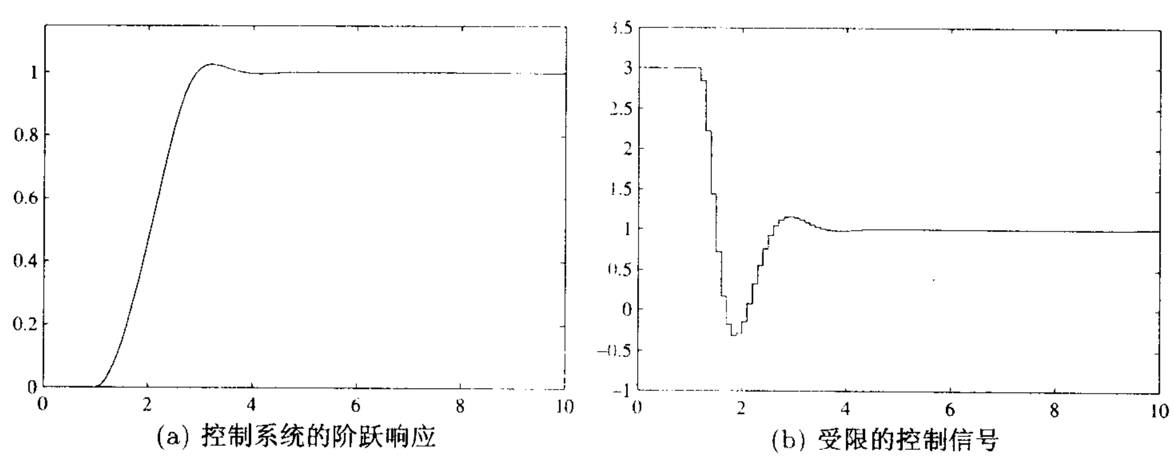
$$[\boldsymbol{y},\boldsymbol{u},\boldsymbol{y}_{\rm m}]=\text{cmpc}(\text{plant},\text{mod},\boldsymbol{q},\boldsymbol{r},M,P,t_{\rm end},s_{\rm p},\boldsymbol{u}_{\rm lim},\boldsymbol{y}_{\rm lim})$$

其中 $\boldsymbol{u}_{\rm lim}=[\boldsymbol{u}_{\rm m},\boldsymbol{u}_{\rm M},\Delta\boldsymbol{u}_{\rm M}]$，$\boldsymbol{y}_{\rm lim}=[\boldsymbol{y}_{\rm m},\boldsymbol{y}_{\rm M},\Delta\boldsymbol{y}_{\rm M}]$。和前面介绍的基于无约束最优化的预测控制和仿真函数相比，可见 cmpc() 函数集设计和仿真于一体，可以直接对系统进行仿真分析，直接得出相应的仿真结果。该函数允许对输入信号和输出信号添加范围约束。

例 6-23 考虑例 6-20 的受控对象模型，若期望系统的控制信号满足 $u(t)\in(-3,3)$，设计预测控制器并通过仿真分析系统的响应。

求解 假设采用默认的 $\boldsymbol{q},\boldsymbol{r}$ 加权，则可以得出 $P=40$，选择不同的控制时域 M，有约束的模型预测控制器可以设计出来，系统的阶跃响应和控制信号如图 6-26 (a)、(b) 所示。可见，控制结果是很理想的，控制信号被有效地限制在预先指定的范围，达到了控制的目的。

```
>> t_f=12; T=0.1; G=poly2tfd(1,conv([1 1],[2 1]),0,1);
   mod=tfd2step(t_f,T,1,G); P=40;M=10; q=1; r=0.1;
   plant=mod; sp=1; tend=10; t=0:T:tend;
   [y,u]=cmpc(plant,mod,q,r,M,P,tend,sp,[-3,3,inf]);
   plot(t,y), figure, stairs(t,u)
```



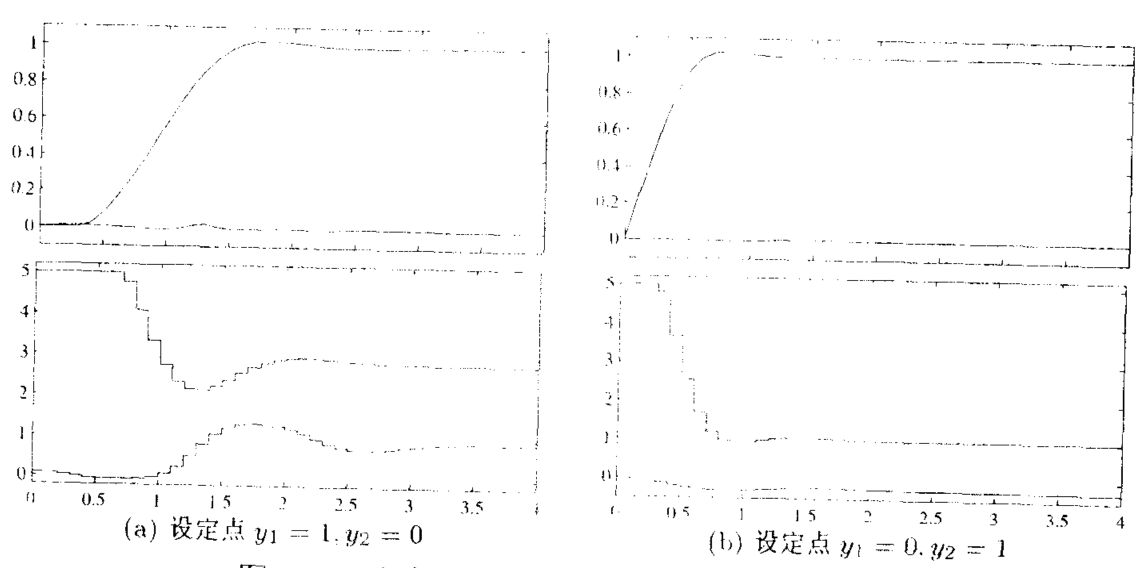
(a) 控制系统的阶跃响应　　(b) 受限的控制信号

图 6-26 有约束的预测控制器的仿真结果

例 6-24 再考虑例 6-21 中给出的多变量系统，假设两路输入信号均满足 $|u_i(t)|\leqslant 5$，试设计出模型预测控制器，并得出仿真结果。

求解 假设采样周期为 $T=0.1$，则可以用类似的方法设计控制器。令两路信号分别单独作用即可得出控制效果。注意输入信号范围的表示方法。这样得出的控制效果和控制信号序列分别如图 6-27 (a)、(b) 所示。

```
>> g11=poly2tfd(0.1134,[1.78,4.48,1],0,0.72);
   g12=poly2tfd(0.924,[2.07,1],0,0);
   g21=poly2tfd(0.3378,[0.361,1.09,1],0,0.3);
   g22=poly2tfd(-0.318,[2.93,1],0,1.29);
   T=0.1; S=tfd2step(20,T,[1 1],g11,g12,g21,g22);
   P=120; M=15; q=[1 1]; r=[0.1 0.1]; plant=S; sp1=[1 0];
   sp2=[0,1]; tend=4; t=0:T:tend;
   [y1,u1]=cmpc(plant,S,q,r,M,P,tend,sp1,[-5,-5,5 5 inf inf]);
   subplot(211); plot(t,y1); subplot(212), stairs(t,u1)
   [y2,u2]=cmpc(plant,S,q,r,M,P,tend,sp2,[-5,-5,5 5 inf inf]);
   figure; subplot(211); plot(t,y2); subplot(212), stairs(t,u2)
```



(a) 设定点 $y_1=1, y_2=0$ (b) 设定点 $y_1=0, y_2=1$

图 6-27 多变量系统的预测控制器的仿真结果

6.5.2 复杂系统的模型预测控制与仿真

较新版本的模型预测工具箱提供了控制器设计的图形用户界面，可以方便地设计出所需的模型预测控制器，并有 Simulink 版的模型预测控制器模块，可以用 Simulink 对复杂的模型预测控制系统进行仿真分析。

在 MATLAB 命令窗口中给出 `mpctool` 命令，则可以打开如图 6-28 所示的图形用户界面，其中，左侧的栏目三个部分分别为 Plant model (受控对象模型)、Controllers (模型预测控制器) 和 Scenarios (控制器仿真工具)。

单击 Import Plant 按钮，则出现一个对话框，允许用户输入受控对象数学模型，如系统的传递函数模型。进入控制器栏目，则出现如图 6-29 所示的对话框栏目，允许输入模型预测控制器的各种参数，如采样周期和两个时域的值。标签

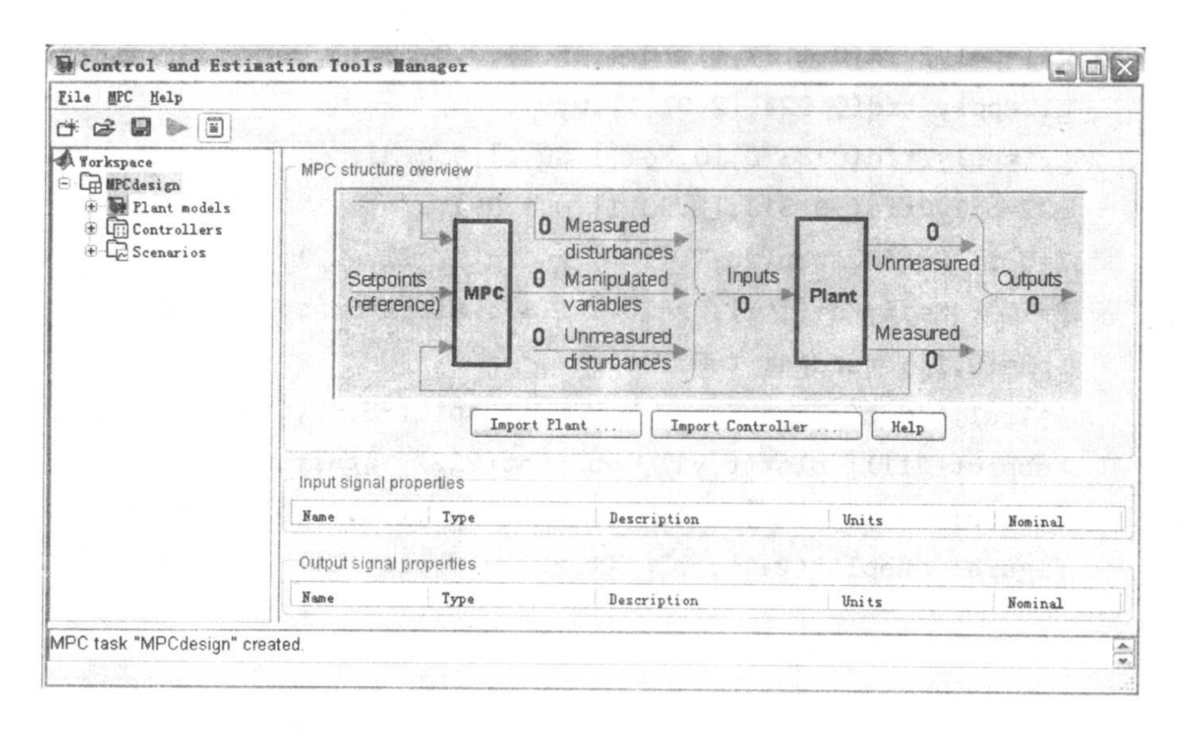


图 6-28 预测控制器设计界面

Constraints 和 Weight Tuning 分别对应控制器设计的约束条件和加权等。

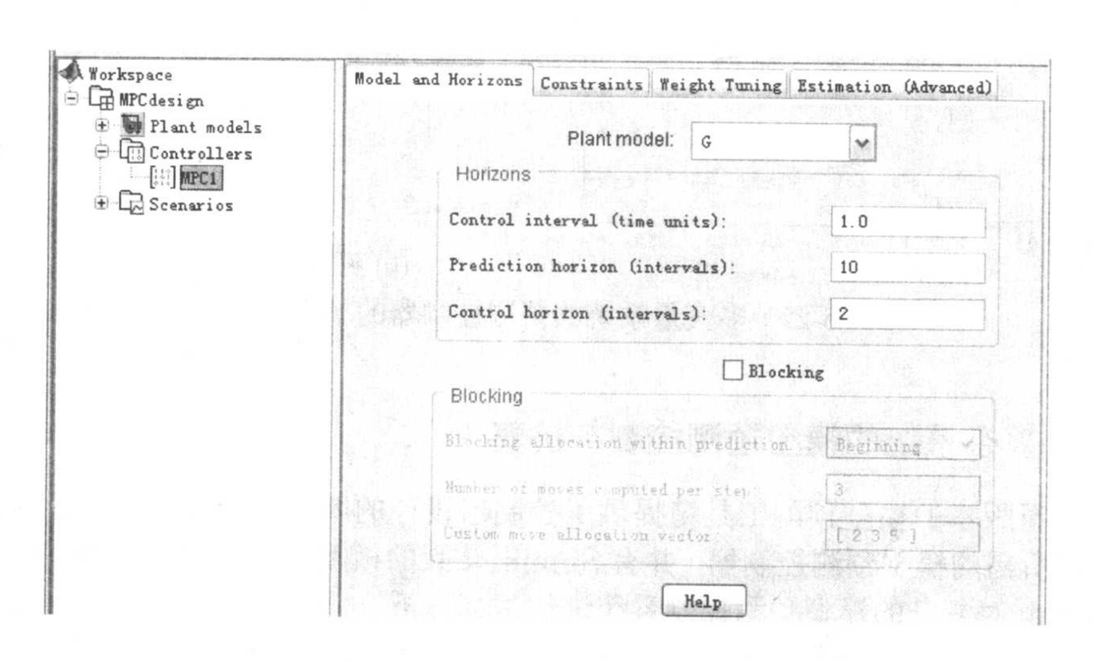


图 6-29 控制器参数设置对话框

描述了受控对象和控制器模型后，则可以用 Scenarios 栏目来对系统进行仿真分析。选中该栏目则出现如图 6-30 所示的对话框，用户可以提供设定点、仿真时间等信息，然后对所设计的模型预测控制器开始仿真分析。

若仿真结果满意，则可以单击控制栏目下的 Export Controller 按钮将控制器

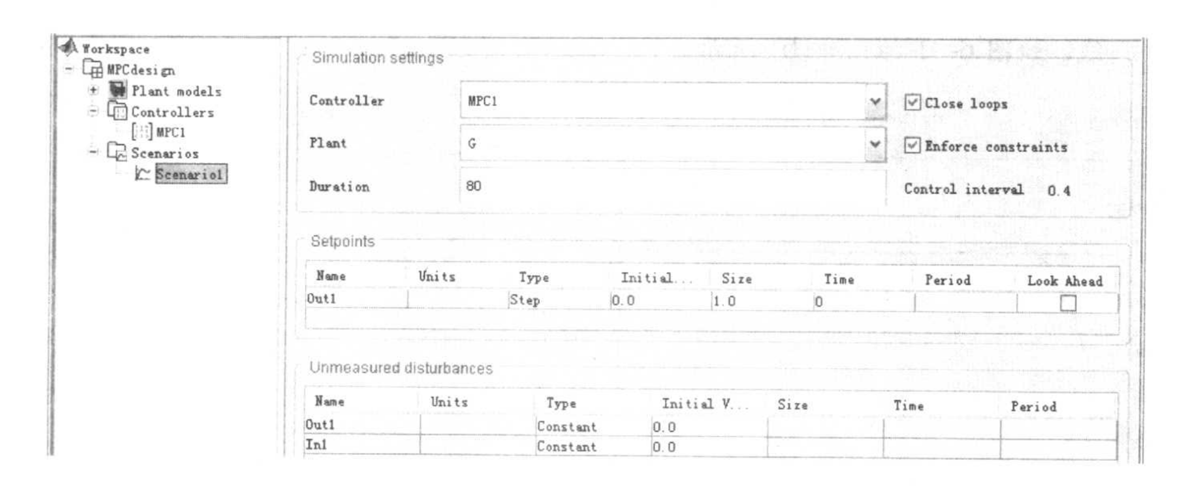


图 6-30 仿真参数设置对话框

输出到 MATLAB 工作空间或文件，以备以后使用。给出 `mpclib` 还将得到可以适用于 Simulink 仿真的模型预测控制器模块，用于复杂系统的设计与仿真。下面通过例子演示模型预测控制器设计与仿真的全过程。

例 6-25 考虑文献 [13] 中给出的变化系统分段受控对象模型，如表 6-5 所示。试设计出可以用于该分段模型控制的模型预测控制器。

表 6-5 受控对象的数学模型

序号	采样区间	对象模型	序号	采样区间	对象模型
1	1-79	$\dfrac{1}{40s^2+10s+1}$	2	8-159	$\dfrac{e^{-2.7s}}{40s^2+10s+1}$
3	160-239	$\dfrac{e^{-2.7s}}{1+10s}$	4	240-319	$\dfrac{1}{10s+1}$
5	320-400	$\dfrac{1}{10s(25s+1)}$			

求解 选择第一模型作为标称模型，则可以用下面的步骤设计模型预测控制器模型：

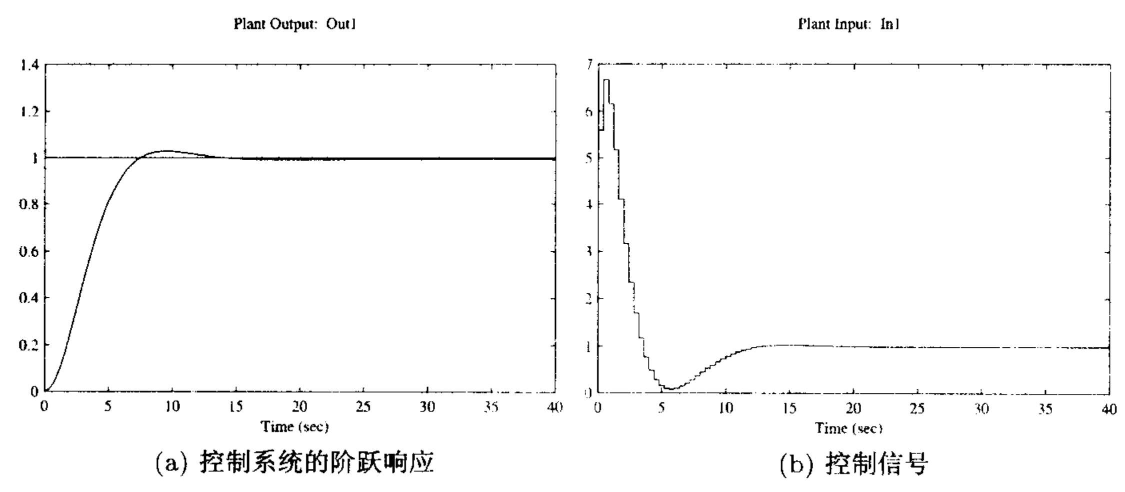
① 输入模型 G=`tf(1,[40,10,1])`。

② 输入 `mpctool` 启动界面，由 Import Plant 按钮读入系统的受控对象模型。

③ 单击左侧的 Controller 栏目，可以如下设置控制器所需参数：在 Model and Horizons 标签下，采样周期 Control interval 为 $T=0.4$，预测时域 Prediction Horizon 设置成 40，控制时域设置为 10；由于暂时不考虑控制约束，所以不修改 Constraints 标签；在 Weight Tuning 标签下，将输入权值设置成 0.1，变化率从默认的 0.1 改变为 0。输出的权值最好设置成 1。

④ 单击左侧 Scenarios 栏目，可以设置设定点等信息，如可以将设定点设置成阶跃信号，使其阶跃时间为 0，幅值为 1。这时可以启动仿真过程得出输入输出曲

线，如图 6-31 (a) 和 (b) 所示。



(a) 控制系统的阶跃响应　　(b) 控制信号

图 6-31　预测控制器的仿真结果

⑤ 选择控制器栏目下的 Export Controller，则可以将控制器模型存成 MATLAB 工作空间中的变量 c6MPC1。

设计了控制器，则可以搭建如图 6-32 所示的分段控制模型，该模型由仿真时间控制模型的选择。这样可以用设计出的模型预测控制器对系统进行控制，得出如图 6-33 (a) 所示的设计效果。

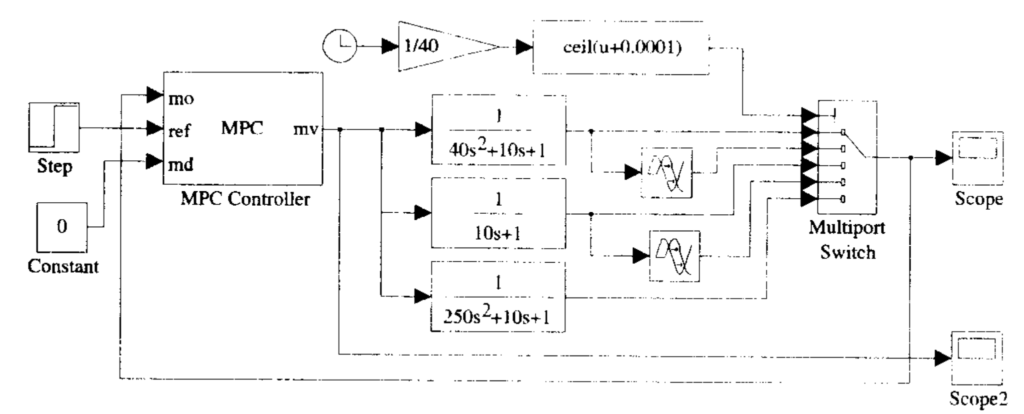


图 6-32　分段系统的预测控制仿真模型 (文件名：c6mmpcm.mdl)

若在设计控制器时选择控制信号的约束 $|u(t)| \leqslant 2$，则可以重新设计出模型预测控制器，在该控制器下系统的响应如图 6-33 (b) 所示。

6.5.3 广义预测控制设计与仿真

广义预测控制 (general predictive control，GPC) 是由英国学者 David Clarke 教授及合作者提出的一种新型控制策略[14,15]，文献 [16] 对广义预测

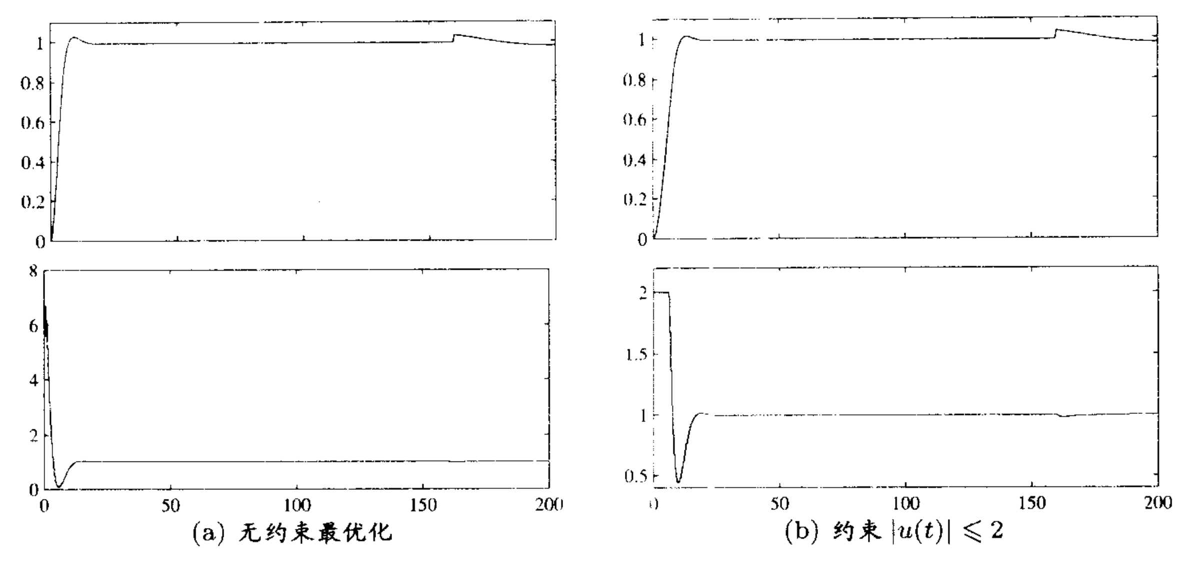


图 6-33 分段系统的预测控制仿真结果

控制理论及应用有较好的介绍。广义预测控制研究的受控对象模型可以表示为

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t) + C(z^{-1})\xi(t) + \eta \tag{6-5-9}$$

该模型的大部分内容和前面描述的完全一致，它的特点是可以处理在模型的输入中添加常数偏差扰动 η 的问题。

广义预测控制的性能指标为

$$J = \min_{u} \mathrm{E}\left\{ \sum_{j=N_1}^{N_2} \Big[y(t+j) - y_{\mathrm{r}}(t+j) \Big]^2 + \sum_{j=1}^{N_u} \lambda \Big[\Delta u(t+j-1) \Big]^2 \right\} \tag{6-5-10}$$

这里，性能指标选择的是一个时域窗口 (N_1, N_2) 内的方差值，属于滚动型性能指标。文献 [17] 给出了较好的基于早期 MATLAB 版本的仿真模型和描述广义预测控制的 S-函数，根据本书的风格修改如下。

```
function [sys, x0,str,ts]=gpc_1a(t,x,u,flag,N1,N2,Nu,r,rho,...
          k_d,B_pocz,A_pocz,P_pocz,alfa,ts)
nA=length(A_pocz); nB=length(B_pocz)-1; k=nA+nB+1;
kp=k*k; kt=kp+1; ktend=kp+k; kf=ktend+1;
kfend=ktend+k; ky=kfend+1; ku=ky+1; kend=ky+k_d;
P=zeros(k,k); x=x(:)';
switch flag
case 0
   sizes = simsizes; % 读入系统变量的默认值
   sizes.NumContStates=0; sizes.NumDiscStates=kend;
   sizes.NumOutputs=1; sizes.NumInputs=2;
   sizes.DirFeedthrough=0; sizes.NumSampleTimes=1;
   sys=simsizes(sizes); str=[]; ts=[ts 0];
```

```
      x0=zeros(1,kend); x0(1:k+1:kp)=P_pocz*ones(k,1);
      x0(kt:kt+nA-1)=A_pocz; x0(kt+nA:ktend)=B_pocz;
   case 2
      Phi=[x(ky),x(kf:kf+nA-2),x(kend),x(kf+nA:kfend-1)];
      P(:)=x(1:kp); P=(1/alfa)*(P-(P*Phi'*Phi*P)/(alfa+Phi*P*Phi'));
      Theta=x(kt:ktend)+Phi*P*(u(2)-Phi*x(kt:ktend)');
      k_M=max([nA+1,nB+k_d]);
      num=[zeros(1,k_d-1),Theta(nA+1:k),zeros(1,k_M-nB-k_d)];
      den=[1,Theta(1:nA),zeros(1,k_M-nA-1)]; h=dstep(num,den,N2);
      for i=1:Nu, Qt(1:N2,i)=[zeros(i-1,1); h(1:N2-i+1)]; end;
      Q=Qt(N1:N2,:); q=[1,zeros(1,Nu-1)]*inv(Q'*Q + r*eye(Nu))*Q';
      [w,xw]=dlsim(rho,1-rho,1,0,u(1)*ones(N2+1,1),u(2));
      A=[1,Theta(1:nA)]; B=Theta(nA+1:k); Bm=[B,0]; Bm=Bm-[0,B];
      Am=[A,0]; Am=Am-[0,A]; Ared=Am(2:nA+2);
      Bred=[zeros(1,k_d-1),Bm]; Y=[u(2),-x(ky),-x(kf:kf+nA-2)];
      U=[x(ku),x(ku:kend),x(kf+nA:kf+nA+nB-1)];
      for i=1:N2
         yp(i)=-Ared*Y'+Bred*U'; Y=[yp(i),Y(1:nA)];
         U=[U(1),U(1:nB+k_d)];
      end
      nu=x(ku)+q*(w(N1+1:N2+1)-yp(N1:N2)');
      sys=[P(:)',Theta,x(ky),x(kf:kf+nA-2), x(kend),...
           x(kf+nA:kfend-1),-u(2), nu, x(ku:kend-1)];
   case 3, sys=x(ku);
   otherwise, sys=[];
   end
```

例 6-26 假设受控对象模型为 $G(s)=\dfrac{1}{2s^2+8s+1}$，且该模型的输出端可能受到 $d=0.5$ 的扰动，试用广义预测控制的方式对该模型进行控制。

求解 为对该系统进行仿真研究，可以建立起如图 6-34 所示的仿真框图。先假设 $d=0$，选择 $N_1=1, N_\mathrm{u}=2$，对不同的 N_2 取值，如 $N_2=3,4,5,6,7$ 进行仿真研究，则可以得出如图 6-35 (a) 所示的输出曲线，从仿真结果可见，$N_2=3$ 时仿真曲线效果不是很好，说明预测 3 步的控制对此例子不适用，故应该增大 N_2 的值，例如选择 $N_2=7$。

图 6-35 (b) 中给出了 $N_2=3$ 和 $N_2=7$ 时的控制信号曲线，可见，只有给定信号突变时，控制信号的要求较大，其他时间控制信号接近 0。

现在假设偏差信号 $\eta=0.5$，则通过仿真可以得出如图 6-36 (a) 所示的控制效果，其中，为了方便比较，同时绘制出 $\eta=0$ 时的控制曲线，可见，虽然模型受到了偏差扰动，控制效果仍然是很理想的。

将受控对象模型修改成 $G_1(s)=\dfrac{3}{3s^2+8s+1}$ 或 $G_2(s)=\dfrac{2s+1}{3s^2+8s+1}$，则得出

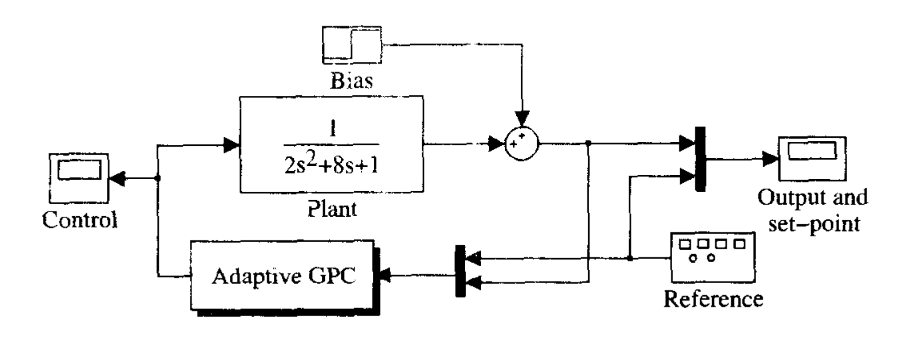
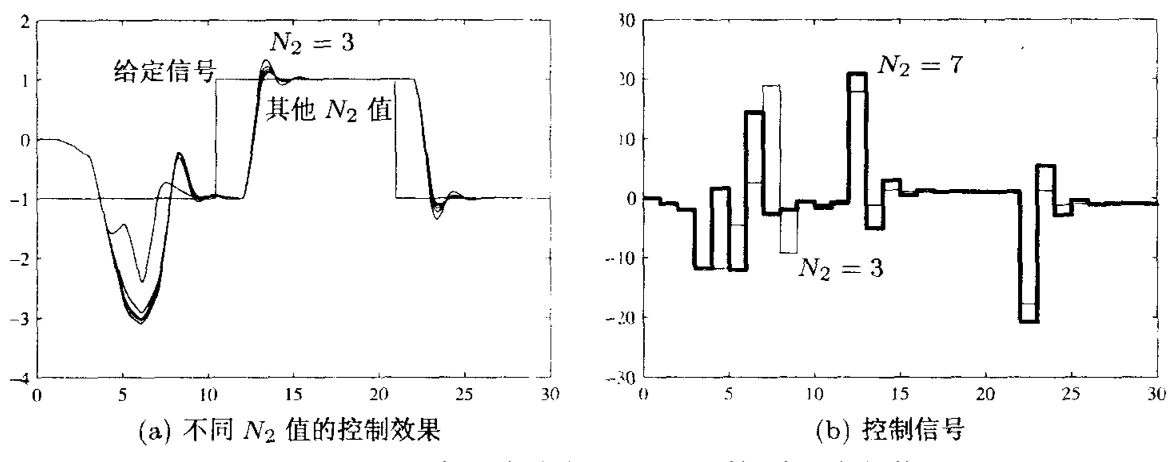
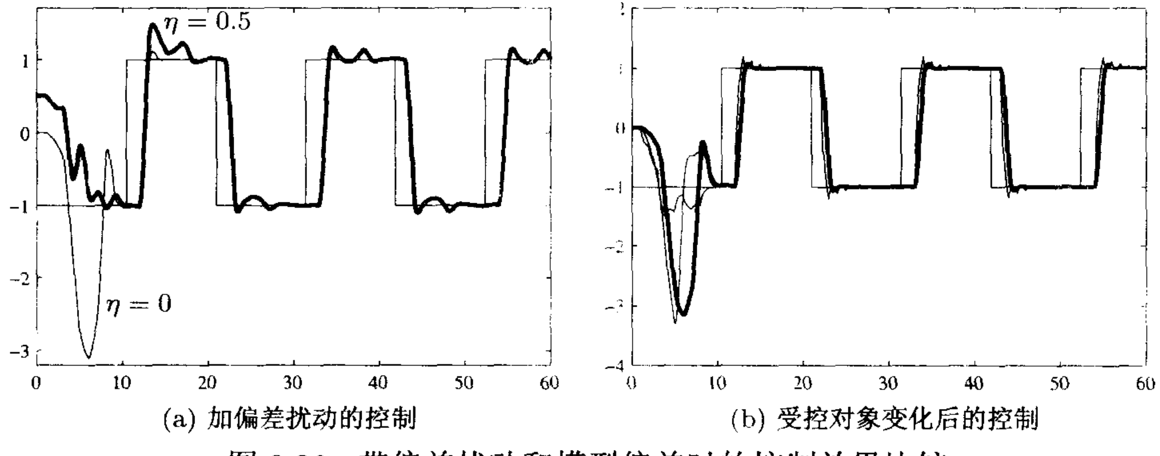


图 6-34 广义预测控制系统的仿真框图 (文件名：c8mgpc1.mdl)



(a) 不同 N_2 值的控制效果 (b) 控制信号

图 6-35 不同窗口宽度的广义预测控制及控制信号



(a) 加偏差扰动的控制 (b) 受控对象变化后的控制

图 6-36 带偏差扰动和模型偏差时的控制效果比较

的控制效果如图 6-36 (b) 所示。可见，虽然受控对象模型有较大的变化，控制效果仍然是较理想的。在得出的曲线中，粗线表示标称模型的响应曲线，其他两条线分别表示 $G_1(s)$ 和 $G_2(s)$ 的响应曲线。

6.6　习题与思考题

1 已知某离散系统的状态方程模型为 $\boldsymbol{x}(t+1)=\begin{bmatrix}0 & 1\\ -0.16 & -1\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}1\\1\end{bmatrix}u(t)$，且 $\boldsymbol{x}^{\mathrm{T}}(0)=[1,-1]$，试求该系统阶跃响应的解析解，并比较数值解。

2 假设受控对象模型为 $G(s)=\left(1+\dfrac{\mathrm{e}^{-s}}{s+1}\right)\dfrac{-0.2s+1}{(s+1)^2}$，如果采用离散 PID 控制器 $u(kT)=1.1646e(kT)+0.2434T\sum\limits_{m=0}^{k}e(mT)+\dfrac{0.0641}{T}\Big\{e(kT)-e[(k-1)T]\Big\}$，试对该系统进行仿真，研究控制效果，通过试凑的方法进一步调整控制器参数，并尝试其他的离散 PID 控制器结构。

3 增量式离散 PID 控制器的数学描述为

$$u(k)=u(k-1)+K_{\mathrm{p}}[e(k)-e(k-1)]+K_{\mathrm{i}}Te(k)+\frac{K_{\mathrm{d}}}{T}[e(k)+e(k-2)-2e(k-1)]$$

试用 Simulink 搭建出该控制器的仿真模型。

4 已知一个离散时间系统的输入输出数据由表 6-6 给出，用最小二乘法辨识出系统的脉冲传递函数模型，并选择合适的模型阶次。

表 6-6　习题 4 实测数据

i	u_i	y_i	i	u_i	y_i	i	u_i	y_i
1	0.9103	0	9	0.9910	54.5252	17	0.6316	62.1589
2	0.7622	18.4984	10	0.3653	65.9972	18	0.8847	63.0000
3	0.2625	31.4285	11	0.2470	62.9181	19	0.2727	68.6356
4	0.0475	32.3228	12	0.9826	57.5592	20	0.4364	60.8267
5	0.7361	28.5690	13	0.7227	67.6080	21	0.7665	57.1745
6	0.3282	39.1704	14	0.7534	70.7397	22	0.4777	60.5321
7	0.6326	39.8825	15	0.6515	73.7718	23	0.2378	57.3803
8	0.7564	46.4963	16	0.0727	74.0165	24	0.2749	49.6011

5 已知某连续系统的阶跃响应数据由表 6-7 给出，且已知系统为二阶系统，其阶跃响应的曲线原型为 $y(t)=x_1+x_2\mathrm{e}^{-x_4t}+x_3\mathrm{e}^{-x_5t}$，试用曲线最小二乘拟合算法拟合出 x_i 参数，从而拟合出系统的传递函数模型。

6 试构造出一组周期为 $N=127$ 的伪随机二进制序列。

7 求 Diophatine 方程的解并验证解的正确性。

① $A(z^{-1})=1-0.7z^{-1}$，$B(z^{-1})=0.9-0.6z^{-1}$，$C(z^{-1})=2z^{-2}+1.5z^{-3}$

② $A(z^{-1})=1+0.6z^{-1}-0.08z^{-2}+0.152z^{-3}+0.0591z^{-4}-0.0365z^{-5}$

$B(z^{-1})=5-4z^{-1}-0.25z^{-2}+0.42z^{-3}$，$C(z^{-1})=1$

表 6-7 习题 5 实测数据

t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$
0	0	1.6	0.2822	3.2	0.3024	4.8	0.3145	6.4	0.3218	8	0.3263
0.1	0.08324	1.7	0.2839	3.3	0.3034	4.9	0.315	6.5	0.3222	8.1	0.3265
0.2	0.1404	1.8	0.2855	3.4	0.3043	5	0.3156	6.6	0.3225	8.2	0.3267
0.3	0.1798	1.9	0.287	3.5	0.3051	5.1	0.3161	6.7	0.3228	8.3	0.3269
0.4	0.2072	2	0.2885	3.6	0.306	5.2	0.3166	6.8	0.3231	8.4	0.3271
0.5	0.2265	2.1	0.2899	3.7	0.3068	5.3	0.3172	6.9	0.3235	8.5	0.3273
0.6	0.2402	2.2	0.2912	3.8	0.3076	5.4	0.3176	7	0.3238	8.6	0.3275
0.7	0.2501	2.3	0.2925	3.9	0.3084	5.5	0.3181	7.1	0.324	8.7	0.3277
0.8	0.2574	2.4	0.2937	4	0.3092	5.6	0.3186	7.2	0.3243	8.8	0.3278
0.9	0.2629	2.5	0.2949	4.1	0.3099	5.7	0.319	7.3	0.3246	8.9	0.328
1	0.2673	2.6	0.2961	4.2	0.3106	5.8	0.3195	7.4	0.3249	9	0.3282
1.1	0.2708	2.7	0.2973	4.3	0.3113	5.9	0.3199	7.5	0.3251	9.1	0.3283
1.2	0.2737	2.8	0.2983	4.4	0.312	6	0.3203	7.6	0.3254	9.2	0.3285
1.3	0.2762	2.9	0.2994	4.5	0.3126	6.1	0.3207	7.7	0.3256	9.3	0.3286
1.4	0.2784	3	0.3004	4.6	0.3133	6.2	0.3211	7.8	0.3258	9.4	0.3288
1.5	0.2804	3.1	0.3014	4.7	0.3139	6.3	0.3214	7.9	0.3261	9.5	0.3289

8 请为受控对象 $(1-1.28z^{-1}+0.49z^{-2})y(t)=(0.5+0.7z^{-1})u(t-1)$ 设计最小方差控制器，并对其进行仿真，观察输出信号是否满意。

9 已知系统模型可以由差分方程 $(1-1.3z^{-1}+0.4z^{-2})y(t)=z^{-2}(1+0.5z^{-1})u(t)+(1-0.65z^{-1}+0.1z^{-2})\epsilon(t)$ 描述，式中 $\epsilon(t)$ 是方差为 0.1 的白噪声信号，试设计最小方差自校正调节器并仿真该系统。

10 已知系统模型为 $y(t)+2.1y(t-1)+1.61y(t-2)+0.531y(t-3)+0.063y(t-4)=2u(t-2)+1.3u(t-3)+0.5\xi(t)+0.5\xi(t-1)+0.2\xi(t-2)$，试构造出该系统的提前两步预报模型，并对方波输入的系统仿真输出信号和预报信号，比较得出的结果。

11 对自校正控制器来说，本章只给出了最小方差自校正调整器的 MATLAB 实现及 Simulink 模块。试仿照该模块将极点配置控制器和系统辨识的功能相结合，编写出极点配置自校正控制器程序与模块，并对慢时变受控对象进行仿真研究，得出期望极点对控制效果的影响。

12 试绘制出极点配置控制器的框图，并用 Simulink 构造出其控制模块。

13 已知某过程控制受控对象为 $G(s)=\dfrac{K\mathrm{e}^{-\theta s}}{\tau s+1}$，参数的标称值为 $K=5,\tau=15,\theta=2$，采样周期为 1，试设计出模型预测控制器，并通过仿真分析控制效果。如果受控对象模型发生了参数变化，例如 $K=10$，试通过仿真的方法研究控制效果。

14 假设某欠阻尼受控对象模型为[11] $G(s)=\dfrac{8611.77}{[(s+0.55)^2+6^2][(s+0.25)^2+15.4^2]}$，

试选择合适的时域，设计出较好的模型预测控制器。如果受控对象变成非最小相位系统，例如分子多项式变成 $28705(-s+0.3)$，试重新研究模型预测控制器设计问题。

15 考虑某多变量传递函数矩阵模型 $G(s)=\begin{bmatrix}\dfrac{12.8\mathrm{e}^{-s}}{16.7s+1} & \dfrac{-18.9\mathrm{e}^{-3s}}{21s+1}\\ \dfrac{6.6\mathrm{e}^{-7s}}{10.9s+1} & \dfrac{-19.4\mathrm{e}^{-3s}}{14.4s+1}\end{bmatrix}$，试设计出相应的模型预测控制器，并仿真控制效果。

16 假设受控对象模型为[16] $y(t)=0.503y(t-1)-0.053y(t-2)+0.017u(t-3)+0.186u(t-4)+0.011u(t-5)+\omega(t)$，其中 $\omega(t)$ 是均值为 0，方差为 0.01 的白噪声，试用广义预测控制对其仿真，得出较好的 N_2 及 N_u 值。

参考文献

[1] 郑大钟. 线性系统理论 (第 2 版) [M]. 北京: 清华大学出版社, 2002
[2] Akaike H. A new look at the statistical model identification [J]. IEEE Transactions on Automatic Control, 1974, AC-19(6):716~723
[3] Ljung L. System identification — theory for the user [M]. Upper Saddle River, N J: PTR Prentice Hall, 2nd , 1999. (清华大学出版社有影印版)
[4] Levy E C. Complex-curve fitting [J]. IRE Transactions on Automatic Control, 1959, AC-4:37~44
[5] 薛定宇. 控制系统仿真与计算机辅助设计 [M]. 北京: 机械工业出版社, 2005
[6] 韩曾晋. 自适应控制系统 [M]. 北京: 机械工业出版社, 1980
[7] 韩曾晋. 自适应控制 [M]. 信息、控制与系统系列教材. 北京: 清华大学出版社, 1995
[8] Åström K J, Wittenmark B. On self-tuning regulators [J]. Automatica, 1973, 9:185~199
[9] 袁震东. 自适应控制理论及其应用 [M]. 上海: 华东师范大学出版社, 1988
[10] Clarke D W, Gawthrop P J. A self-tuning controller [J]. Proceedings IEE, Part D, 1975, 122:929~934
[11] 席裕庚. 预测控制 [M]. 北京: 国防工业出版社, 1993
[12] The MathWorks Inc. Model predictive control toolbox user's manual [Z], 2005
[13] Richalet J. Model predictive heuristic control: applocations to industrial processes [J]. Automatica, 1978, 14(5):413~428
[14] Clarke D W, C Mohtadi, P S Tuffs. Generalized predictive control — Part I. The basic algorithm [J]. Automatica, 1987, 23:137~148
[15] Clarke D W, C Mohtadi, P S Tuffs. Generalized predictive control — Part II. Extensions and interpretations [J]. Automatica, 1987, 23:149~160
[16] 王伟. 广义预测控制理论及其应用 [M]. 北京: 科学出版社, 1998
[17] Mościński J, Ogonowski Z. Advanced control with MATLAB and Simulink [M]. London: Ellis Horwood, 1995

第 7 章 智能计算问题的计算机求解

智能控制的概念是由美国 Purdue 大学著名学者傅京孙 (King-Sun Fu) 教授于 1971 年首先提出的[1]，认为智能控制是人工智能与自动控制的交集，主要强调人工智能中仿人的概念与自动控制的结合[2]。“智能控制”至今无统一的定义，文献 [3] 中给出了一种合理的定义：智能控制是一类无需人的干预就能够独立地驱动智能机器实现其目标的自动控制。与传统的控制理论相比，智能控制对于环境和任务的复杂性有更大的适应度。目前几种被广泛认可的智能控制形式包括专家系统、模糊控制、人工神经网络控制、自学习控制等。

本书将在 7.1 节首先介绍经典集合论问题的求解，然后引入模糊集合与模糊逻辑的概念，介绍模糊逻辑计算与模糊推理和几种常用的模糊控制器形式，并介绍模糊逻辑控制器的设计方法及基于 MATLAB 工具的建模与仿真方法。7.2 节将首先介绍人工神经网络的结构与求解、用神经网络拟合数据的方法及技术细节，然后介绍基于 MATLAB 的神经网络模型预测控制器设计及仿真方法。7.3 节将介绍遗传算法在最优化问题求解及最优控制器设计中的应用，然后介绍基于粒子群优化算法的最优化问题求解方法，并介绍应用这些方法的全局最优化方法及控制器设计问题求解。7.4 节将引入迭代学习控制的理论及相关问题 MATLAB 求解。

7.1 模糊逻辑与模糊控制

7.1.1 经典可枚举集合论问题及 MATLAB 求解

集合论是现代数学的基础。所谓集合，就是一些事物的全体，而其中每一个事物均称为集合中的一个元素。若事物 a 是集合 A 中的一个元素，则记 $a \in A$，称为 a 属于 A。若 b 不是 A 集合中的元素，则记 $b \notin A$。所谓可枚举集合，是指该集合中的所有元素均可以一一列出。在 MATLAB 中用向量或单元数组的形式就可以表示这样的集

合。例如，下面的语句均可以表示可枚举集合。

```
>> A=[1 2 3 5 6 7 9 3 4 11]  % 数字构成的集合，可以有重复元素
   B={1 2 3 5 6 7 9 3 4 11}  % 上述集合的单元数组表示方法，二者等价
   C={'ssa','jsjhs','su','whi','kjshd','kshk'}  % 字符串集合
```

MATLAB 语言提供了集合定义与基本运算函数。在表 7-1 中列出了进行集合运算的函数及解释，用这些函数可以对集合进行操作，这些函数还可以嵌套使用，建立较复杂的集合运算。这些函数不能用于符号表达式的集合运算。

表 7-1 MATLAB 下集合运算的函数

运算名称	MATLAB 语句	集合运算描述
并集运算	A=union(B,C)	求两个集合 B 和 C 的并集，数学记号为 $A=B\bigcup C$，集合运算后的结果被重新排序
差集运算	A=setfiff(B,C)	求两个集合 B 和 C 的差集，记作 $A=B\backslash C$，即从集合 B 中剔除 C 中的元素后剩下的元素，结果被重新排序
交集运算	A=intersect(B,C)	求两个集合 B 和 C 的交集，即 $A=B\bigcap C$，结果被重新排序
异或运算	A=setxor(B,C)	求两个集合 B 和 C 的异或运算，即从 $B\bigcup C$ 中剔除 $B\bigcap C$，数学表示为 $A=(B\bigcup C)\backslash(B\bigcap C)$，结果被重新排序
惟一运算	A=unique(B)	将 B 集合中的重复元素剔除，结果被重新排序
属于判定	key=ismember(a,B)	判定 a 是否为集合 B 中的元素，如果是则返回 key 值为 1，否则返回 0，记作 key=$a\in B$。其实，在属于关系中，a 也可以为矩阵，这时返回的 key 为和 a 一样维数的矩阵，在满足属于关系的元素处为 1，否则为 0

例 7-1 假设给定3个集合 $A=\{1,4,5,8,7,3\}$,$B=\{2,4,6,8,10\}$,$C=\{1,7,4,2,7,9,8\}$，试演示集合的各种运算，并验证它们满足交换律 $(A\bigcup B)\bigcap C=(A\bigcap C)\bigcup(B\bigcap C)$。

求解 由给出的条件可以立即输入已知的 A,B,C 这 3 个集合，然后调用集合运算的命令即可以得出如下的结果。

```
>> A=[1,4,5,8,7,3]; B=[2,4,6,8,10]; C=[1,7,4,2,7,9,8]; % 集合定义
   D=unique(C), E=union(A,B), F=intersect(A,B)  % 各种集合运算
```

则可以得出集合运算结果 $D=[1,2,4,7,8,9]$，$E=[1,2,3,4,5,6,7,8,10]$，$F=[4,8]$。给出如下命令，则可以发现交换律左侧的集合与右侧的集合求差集，得出的结果为空集，由此验证了交换律的正确性。

```
>> G=setdiff(intersect(union(A,B),C),...
              union(intersect(A,C),intersect(B,C)))
```

命令 G=ismember(A,B) 可以得出属于关系向量为 $G=[0,1,0,1,0,0]$。由得出的结果看，集合 A 中的第 2 和第 4 元素属于集合 B，因为这些位置处测试结果的值为 1。所以，可以用 H=A(ismember(A,B)) 语句提取出集合 A 中属于 B 的元素，亦即得出 $A\bigcap B$，为 $H=[4,8]$。

例 7-2 假设集合 A 和 B 为字符串组，其中

A ={'skhsak','ssd','ssfa'}，B ={'sdsd','ssd','sssf'}，

试求它们的并集与交集，令 C={'jsg','sjjfs','ssd'}，试验证结合律

$$\left(A\bigcap B\right)\bigcup\left(C\bigcap B\right)=\left(A\bigcup C\right)\bigcap B$$

求解 字符串构成的集合可以用单元数组的形式表示，也可以进行集合运算，所以直接用下面的语句求出它们的交集和并集为

```
>> A={'skhsak','ssd','ssfa'}; B={'sdsd','ssd','sssf'};
   F=union(A,B), D=intersect(A,B)
   C={'jsg','sjjfs','ssd'};  % 可以由下面的集合运算验证结合律
   E=setdiff(union(intersect(A,B),...
                 intersect(C,B)),intersect(union(A,C),B))
```

得出的结果为 F ={'sdsd','skhsak','ssd','ssfa','sssf'}，D={'ssd'}，且得出集合 E 为空集。

子集与集合包含等概念是集合论中很重要的概念。所谓集合包含即集合 A 中所有的元素均为集合 B 的元素，记作 $A\subseteq B$，称为 B 包含 A，又称 A 是 B 的子集。若 $B\backslash A$ 非空，则称严格包含，记作 $A\subset B$。MATLAB 中并未直接提供集合包含或子集的函数，但可以通过下面的命令判定包含和严格包含。

key=all(ismember(A,B))，key= 1 则 $A\subseteq B$，即判定 A 所有元素均属于 B

key=all(ismember(A,B))&(length(setdiff(B,A))>0)，key= 1 则 $A\subset B$

例 7-3 考虑例 7-2 中的集合 E, F，试判定 $F\subset E$ 是否满足，并由集合 A 验证集合的自反律，亦即 $A\subseteq A$。

求解 可以用下面的语句进行判定，得出 key 为 1。

```
>> E=union(A,B); F=intersect(A,B); key=all(ismember(F,E))
```

事实上，$F=A\bigcup B$，$E=A\bigcap B$，所以当然 $E\subset F$。还可以验证 $A\subseteq A$，亦即自反律，由下面的语句可以直接得出相应的结果。

```
>> key=all(ismember(A,A)) & (length(setdiff(A,A))>0); % A⊄A
   key1=all(ismember(A,A)); [key,key1]  % A⊆A 当然成立
```

7.1.2 模糊集合

由经典集合论可见，一个事物 a 要么属于集合 A，要么不属于集合 A，没有其他的属于关系。在现实生活中，经常会出现模糊的概念，亦即某一事物 a 以一定程度属于集合 A，该思想是模糊集合的基础。

模糊集合的概念是控制论专家 Lotfi A Zadeh 教授于 1965 年提出的[4]。目前模糊逻辑已经广泛地应用于理、工、农、医各种各样的领域[5]。在自动控制领域中模糊控制也是很有吸引力的研究方向。

在本书前面的介绍中实际上也使用了模糊的概念，例如变步长方法中关于误差的描述是当“误差较大时 ……”，只不过在实际处理时没有使用模糊的方法，而是直接使用了确定性方法解决问题。

这里不加解释地直接引入文献 [6] 给出的示意图来表示精确性与意义性，如图 7-1 所示。可以看出，现实世界中的事物并非都是越精确越好。Zadeh 教授指出，当问题复杂性增加时，精确的描述将失去意义，而有意义的描述将失去精度。

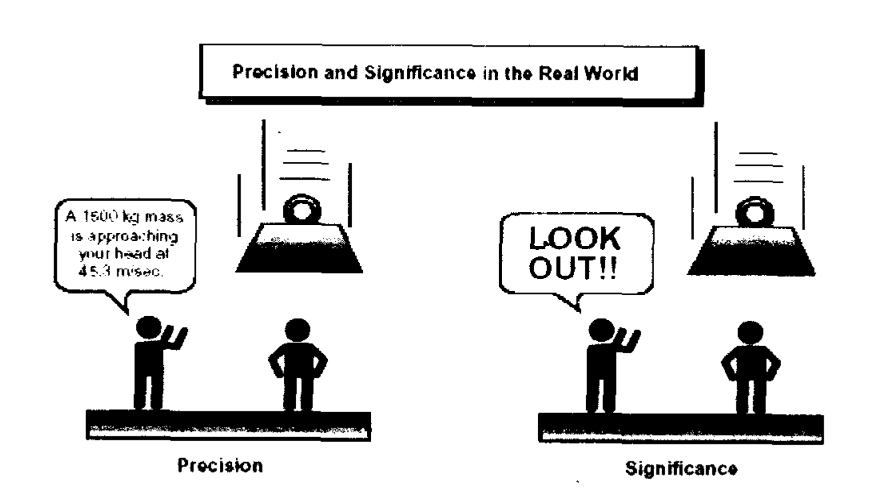


图 7-1 在现实世界中精确性与意义性示意图 (文献 [6])

7.1.3 隶属度与模糊化

所谓隶属度即某个元素 x 属于集合 A 的测度或程度。在经典集合理论中，元素 a 要么属于 A，要么不属于 A，所以对应的隶属度分别为 1 或 0。模糊集合的引入使得这样的属于关系变得柔和了。进一步引入了隶属度函数 $f(x)$，该函数满足 $0 \leqslant f(x) \leqslant 1$。这里将介绍几种常用的隶属度函数及其 MATLAB 求解方法。

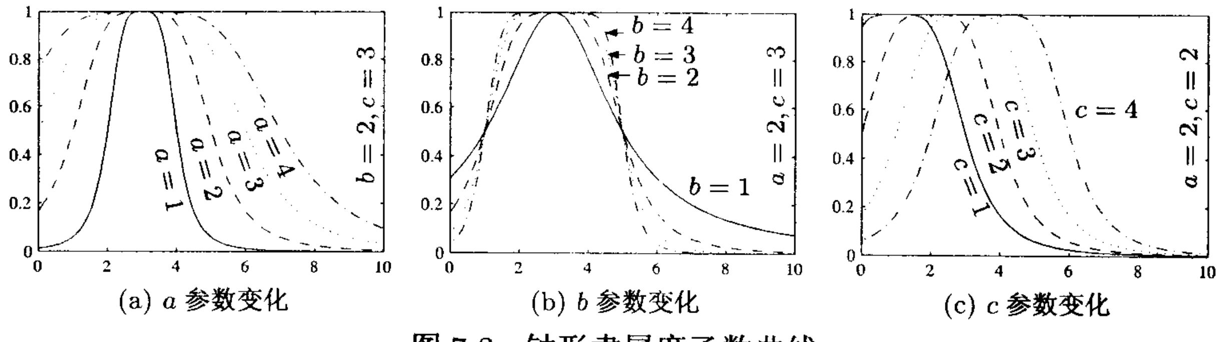
1. 钟形隶属度函数

钟形隶属度函数的数学表达式为 $f(x)=\dfrac{1}{1+|(x-c)/a|^{2b}}$。MATLAB 模糊逻辑工具箱中提供了函数 gbellmf()，可以求出隶属度函数的值。该函数的调用格式为 y=gbellmf(x,[a,b,c])，其中，x 为任意给定的自变量值。调用此函数

则可以求出 $\boldsymbol{x}$ 处的隶属度函数值 $\boldsymbol{y}$。

例 7-4 用绘制出不同参数组合下的钟形隶属度函数曲线。具体的方法是，先选定 $\boldsymbol{x}$ 向量，分别改变 a,b,c 的值，则可以得出如图 7-2 所示的隶属度函数曲线，从得出的曲线可以观察出隶属度函数对 a,b,c 参数的依赖关系。

```
>> x=[0:0.05:10]'; y=[]; a0=1:5; b=2; c=3; y1=[]; y2=[];
   for a=a0, y=[y gbellmf(x,[a,b,c])]; end
   a=1; b0=1:4; c=3; for b=b0, y1=[y1 gbellmf(x,[a,b,c])]; end
   a=2; b=2; c0=1:4; for c=c0, y2=[y2 gbellmf(x,[a,b,c])]; end
   plot(x,y); figure; plot(x,y1); figure; plot(x,y2)
```



(a) a 参数变化　(b) b 参数变化　(c) c 参数变化

图 7-2　钟形隶属度函数曲线

从得出的曲线形状可以看出，当其他参数不变，只修改 a 值时，a 值越小则曲线形状越窄；c 参数只能用于平移曲线，不改变曲线的形状；b 参数增大将增加上升段和下降段的陡度。可以通过这些参数的组合有意识地得出合适的隶属度函数。

2. Gauss 隶属度函数

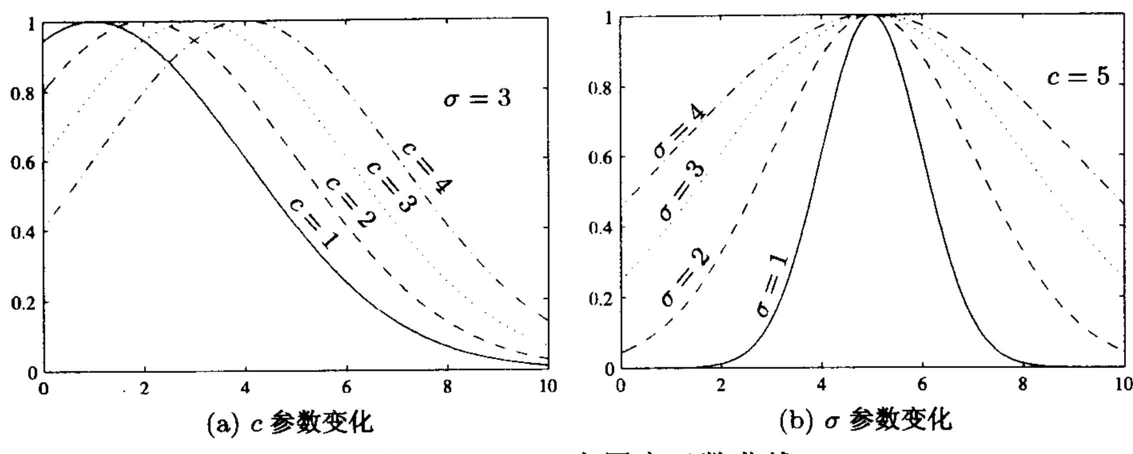
Gauss 隶属度函数的数学表达式为 $f(x)=\mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}}$。MATLAB 模糊逻辑工具箱中提供了 **gaussmf()** 函数，可以求取 Gauss 隶属度的值。该函数的调用格式为 $\boldsymbol{y}$**=gaussmf(**$\boldsymbol{x}$**,[**σ**,**c**])**。

例 7-5 不同 c 和 σ 参数的 Gauss 隶属度函数可以通过下面的语句绘制出来，如图 7-3 所示。可以看出，当 c 变化时，隶属度函数曲线形状不变，只作左右平移，σ 增大时曲线变宽。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; s=3; y1=[]; sig0=1:4;
   for c=c0, y=[y gaussmf(x,[s,c])]; end
   c=5; for sig=sig0, y1=[y1 gaussmf(x,[sig,c])]; end;
   plot(x,y); figure; plot(x,y1)
```

3. Sigmoid 型隶属度函数

Sigmoid 型隶属度函数的数学表达式为 $f(x)=\dfrac{1}{1+\mathrm{e}^{-a(x-c)}}$，该隶属度可以



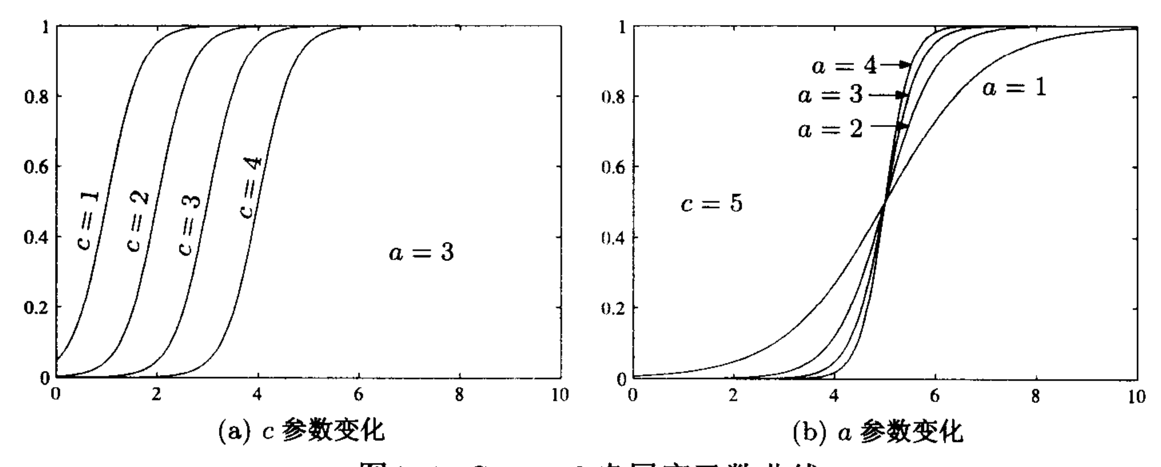
(a) c 参数变化　　(b) σ 参数变化

图 7-3　Gauss 隶属度函数曲线

用 MATLAB 函数 `sigmf()` 求出 `y=sigmf(x,[a,c])`。

例 7-6 Sigmoid 函数在 a 和 c 变量的不同取值下隶属度函数形状如图 7-4 所示。可见，当 c 参数增加或减小时，Sigmoid 函数向右或向左进行平移，而隶属度函数的形状不变；当 a 参数增大或减小时，曲线变得更陡或更平缓。另外应该注意，该函数是单值的，故适用于描述最右侧模糊集合的隶属度，最左侧集合的隶属度可以由 $1-f(x)$ 来表示。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; a=3; y1=[]; a0=1:2:7;
   for c=c0, y=[y sigmf(x,[a,c])]; end
   c=5; for a=a0, y1=[y1 sigmf(x,[a,c])]; end;
   plot(x,y); figure; plot(x,y1)
```



(a) c 参数变化　　(b) a 参数变化

图 7-4　Sigmoid 隶属度函数曲线

4. 隶属度函数的图形编辑界面

MATLAB 模糊逻辑工具箱中提供了隶属度函数的编辑界面。在 MATLAB 提示符下输入 `mfedit` 命令就可以打开隶属度函数编辑界面，如图 7-5 所示。其中给出了 3 个隶属度函数的原型，用户可以通过界面中的选项设置各种隶属度函

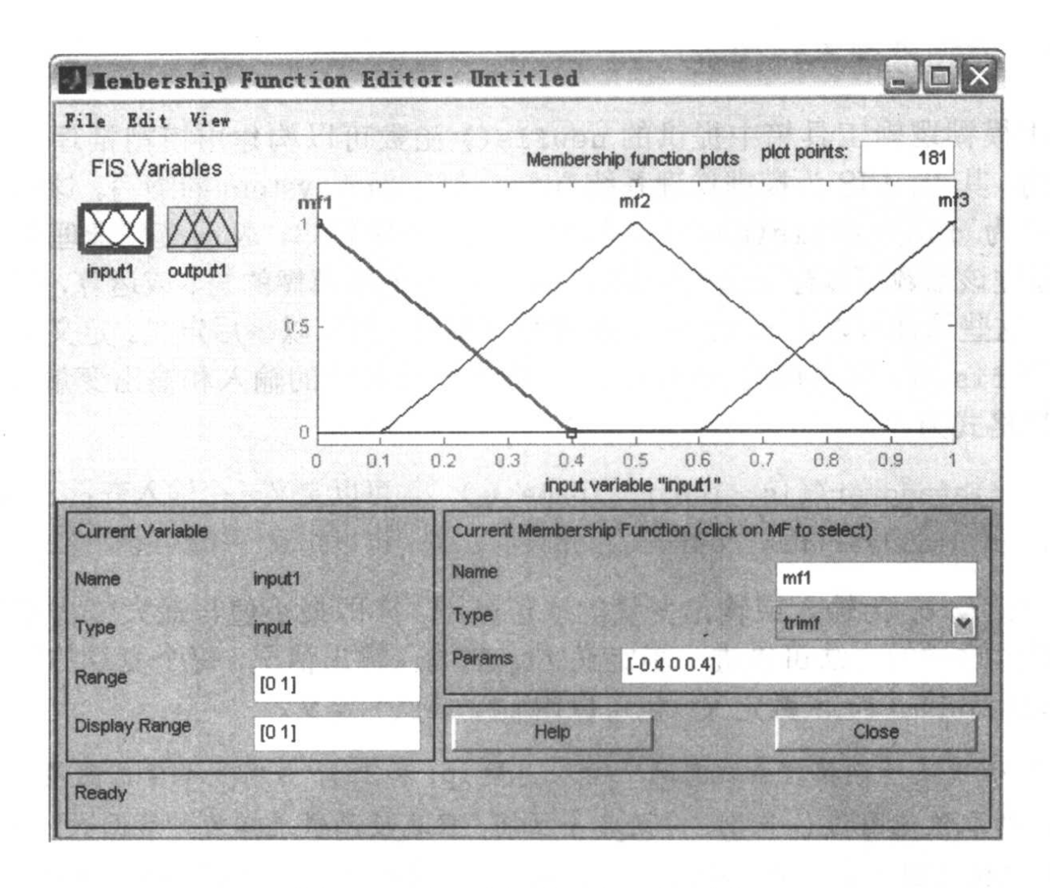


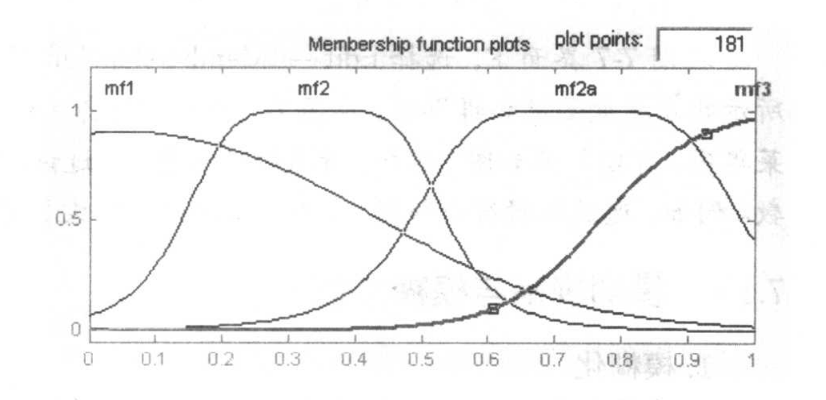
图 7-5 隶属度函数图形编辑界面

数，可以由对话框右下栏目中的内容对当前隶属度函数的形状和参数进行编辑，也可以通过鼠标在隶属度函数示意图上可视地修改隶属度函数的参数。

如果想再添加一个隶属度函数，则可以选择 Edit → Add custom MF 菜单，如图 7-6 (a) 所示，设置完成后就可以在编辑区域添加一个隶属度函数，对这个新添加的隶属度函数可以按前面的方式进行修改，例如可以改变成如图 7-6 (b) 所示的形式。

Edit View
Undo Ctrl+Z
Add MFs...
Add Custom MF...
Remove Selected MF
Remove All MFs
FIS Properties... Ctrl+1
Rules... Ctrl+3

(a) Edit 菜单



(b) 修改后的隶属度函数

图 7-6 隶属度函数的编辑结果

7.1.4 模糊推理系统建立

用模糊逻辑工具箱中提供的 newfis() 函数可以构建出模糊推理系统的数据结构。其中，FIS 为模糊推理系统 fuzzy inference system 的缩写。该函数的调用格式为 fis=newfis(name)，其中，name 为字符串，表示模糊推理系统的名称，通过该函数可以建立起结构体 fis，其内容包括模糊的与、或运算，解模糊算法等，这些属性可以由 newfis() 函数直接定义，也可以事后定义。定义了模糊推理系统 fis 后，可以调用 addvar() 函数来添加系统的输入和输出变量。该函数的调用格式为

fis=addvar(fis,'input',iname,$v_{\rm i}$)	可以定义一个输入变量 iname
fis=addvar(fis,'output',oname,$v_{\rm o}$)	可以定义一个输出变量 oname

其中，$v_{\rm i}$ 及 $v_{\rm o}$ 为输入或输出变量的取值范围，亦即最小值与最大值构成的行向量。通过这样的方法可以进一步定义 fis 的输入输出情况，每个变量的隶属度函数可以用 addmf() 函数定义，也可以用 mfedit() 定义。

例 7-7 假设某模糊推理系统有两个输入变量 ip_1 和 ip_2，并有一个输出变量 op，且假设 ip_1 的取值范围为 $(-3,3)$，分为 3 个区间，隶属度函数选择为钟形函数；输入信号 ip_2 的取值范围为 $(-5,5)$，分为 3 个区间，隶属度函数选择为 Gauss 型函数；输出信号 op 的取值范围为 $(-2,2)$，隶属度函数为 Sigmoid 型函数，则可以用下面的语句构造模糊推理系统原型，并用 fuzzy() 函数编辑此模糊推理系统。

```
>> fff=newfis('c7mfis');                          % 建立模糊推理系统模型
   fff=addvar(fff,'input','ip1',[-3,3]);  % 定义第一路输入信号
   fff=addvar(fff,'input','ip2',[-5,5]);  % 定义第二路输入信号
   fff=addvar(fff,'output','op',[-2,2]);  % 定义输出信号
   fuzzy(fff)             % 用 fuzzy() 函数可视地编辑模糊推理系统
```

由 fuzzy() 函数可以打开模糊推理系统的程序界面，如图 7-7 所示。

在图 7-7 界面下，选择 Edit → Membership functions 菜单项，可以打开如图 7-5 所示的隶属度函数编辑界面。在图 7-7 界面上选择 ip_1 图标，再选择 Edit → Add MFs 菜单项，可以打开如图 7-8 (a) 所示的对话框，通过该对话框定义各个信号的隶属度函数。例如，通过编辑得出如图 7-8 (b) 所示的输出隶属度函数。

7.1.5 模糊规则与模糊推理

1. 模糊化

若将某信号用三个隶属度函数表示，则一般对应的物理意义是“很小”、“中等”与“较大”，若分为 5 段，则可以表示为“很小”、“较小”、“中等”、“较大”和“很大”，一个精确的信号可以通过这样一组隶属度函数模糊化，变成模糊信号。

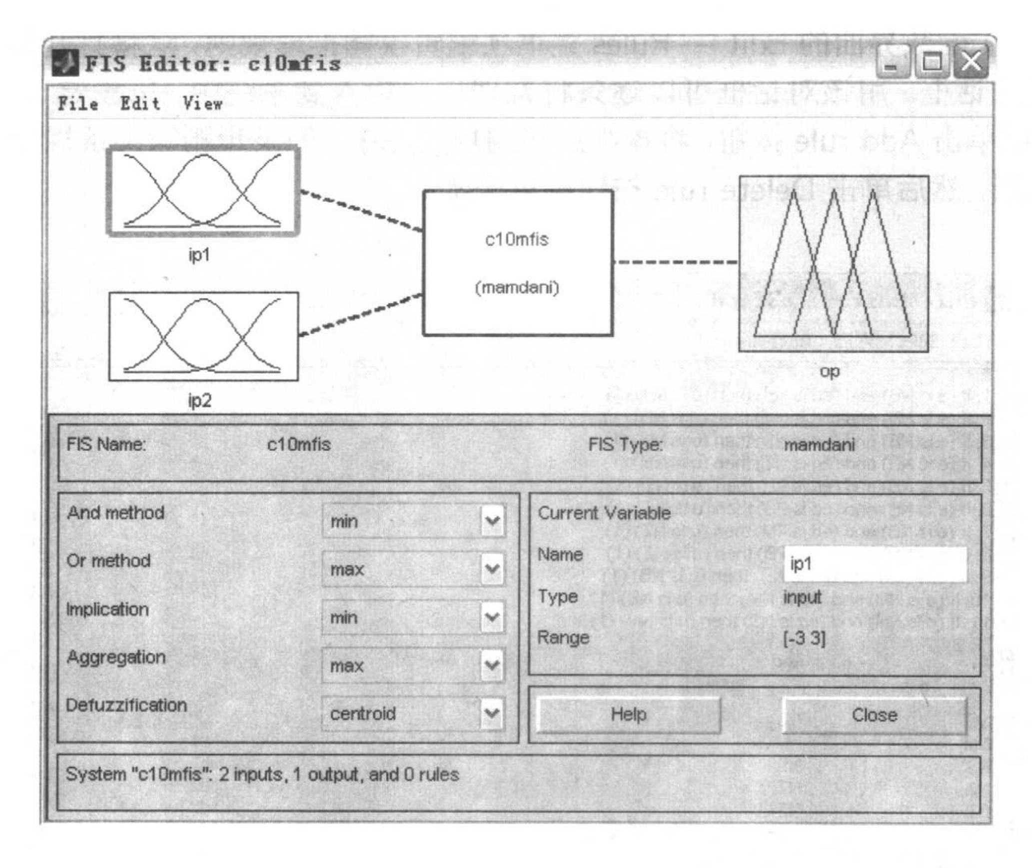
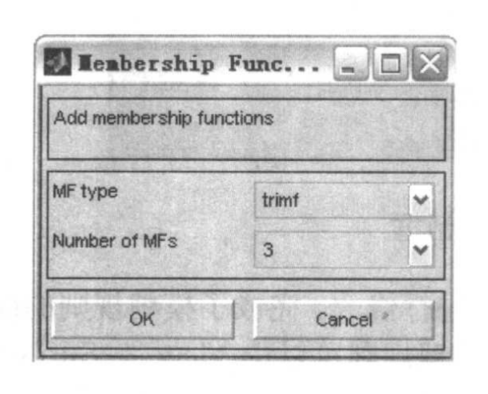
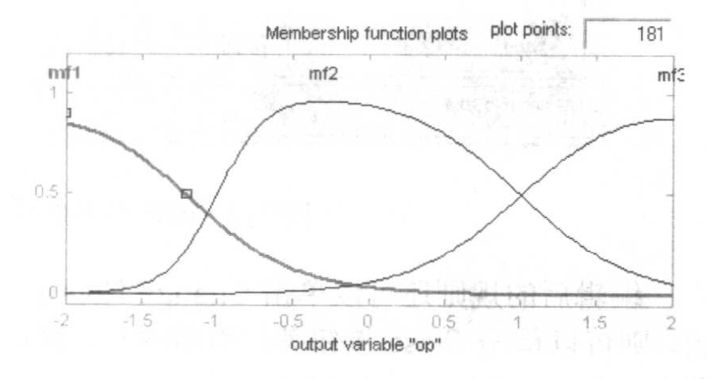


图 7-7 模糊推理系统编辑界面



(a) 隶属度函数设置对话框



(b) 修改后的输出变量隶属度函数

图 7-8 隶属度函数的编辑结果

2. 模糊规则

如果将多路信号均模糊化，则可以用 if，else 型语句表示出模糊推理关系。例如，若输入信号 ip_1 “很小”，且输入信号 ip_2 “较大”，则设置 “较大” 的输出信号 op，这样的推理关系可以表示成

if ip_1 为 “很小” **and** ip_2 为 “很大”，**then** op=“很大”

模糊推理规则可以通过 ruleedit() 函数生成的界面来设定，也可以从

mfedit() 函数界面的 Edit → Rules 菜单项编辑模糊推理规则，这将打开如图 7-9 所示的对话框。用该对话框可以逐条将推理规则输入到系统中，每设定一条规则后，可以单击 Add rule 按钮，将规则添加到规则库中。如果想删除某条规则，则选中该规则，然后单击 Delete rule 按钮即可删除。

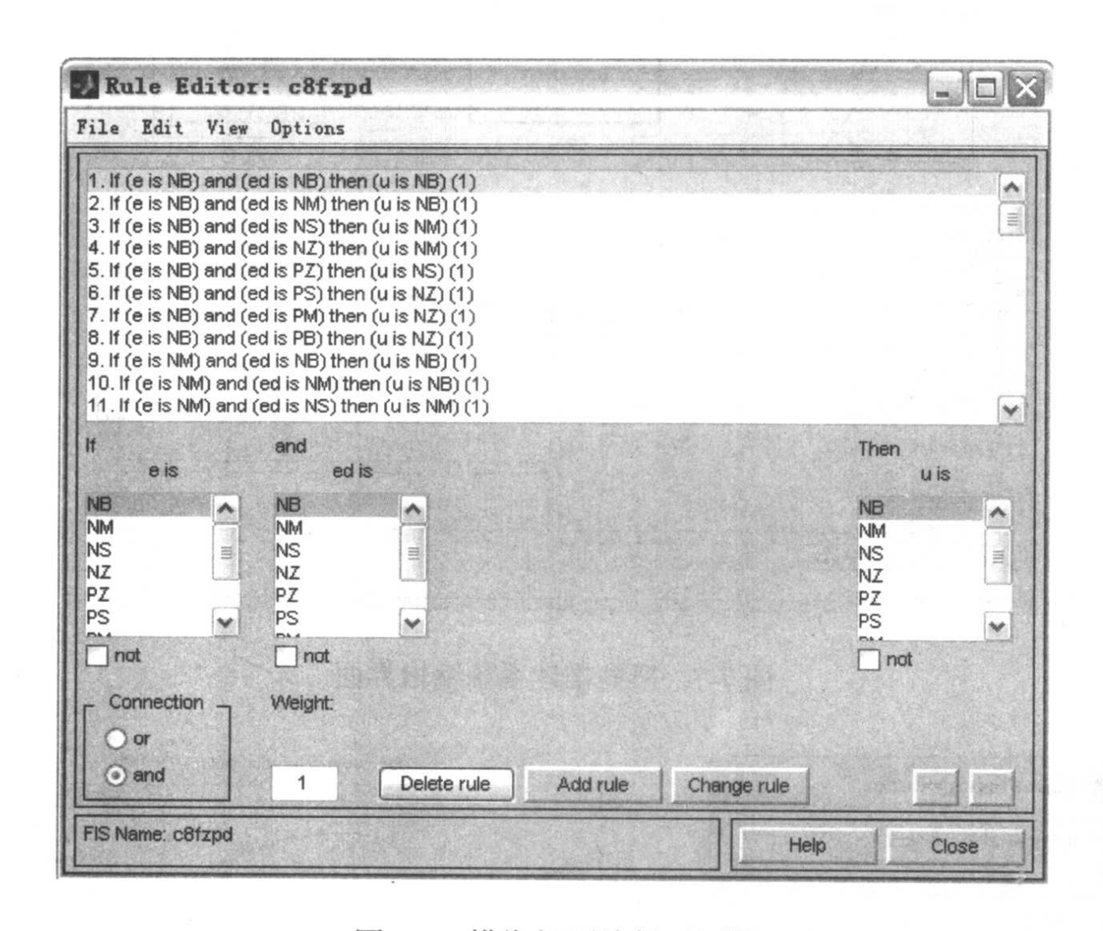


图 7-9 模糊规则编辑对话框

编辑后的规则还可以单击 Change Rule 按钮进行修改。完成了模糊规则的编辑，则可以单击 Close 按钮关闭编辑窗口。模糊推理规则可以由 View → Surface 菜单项进行处理，得出如图 7-10 所示的三维图形，表明从输入信号到输出信号的映射关系。

模糊规则还可以更简单地用数据向量表示，多行向量可以构成多条模糊规则矩阵。每行向量有 $m+n+2$ 个元素，m,n 分别为输入变量和输出变量的个数，其中前 m 个元素表示输入信号的隶属度函数序号，次 n 个元素对应输出信号的隶属度函数序号，第 $m+n+1$ 表示输出的加权系数，最后一个元素表示输入信号的逻辑关系，1 表示逻辑"与"，2 表示逻辑"或"。例如对图 7-9 中的第 3 条逻辑关系来说，若用数据向量的形式可以表示为 $[3,2,1,1,1]$。

若用前面的规则生成一个规则矩阵 $\boldsymbol{R}$，则可以由 `fis=addrule(fis,R)` 直接补加到模糊推理系统 `fis` 原有的规则后面。

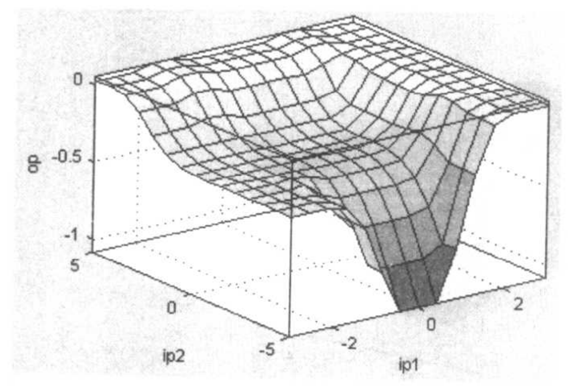



图 7-10 模糊规则的三维表面图

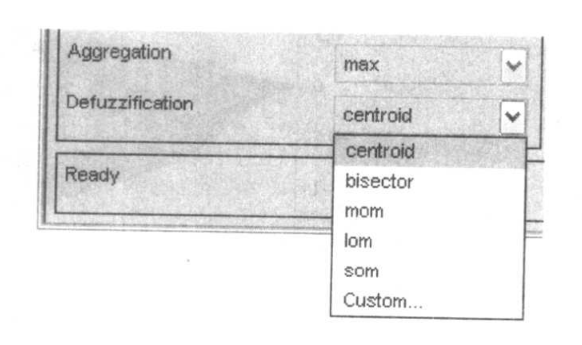


图 7-11 解模糊列表框

3. 解模糊化

通过模糊推理可以得出模糊输出量 op，此模糊量可以通过指定的算法精确化，亦称解模糊化 (defuzzification)。模糊逻辑工具箱提供了多种解模糊化的算法，可以由如图 7-7 所示的对话框 Defuzzifications 栏目，亦即对话框中如图 7-11 所示的部分选择解模糊化算法。

按照上述的方式就可以建立起模糊推理系统的数据结构。可以由 File 菜单对模型进行处理，例如用 File → Export → To Disk 菜单项将其存成文件，后缀名为 fis。用户可以将前面编辑的模糊推理系统存储成 c7mfis.fis 文件。该工作还可以通过 `writefis()` 函数完成。还可以由 File → Export → To Workspace 菜单项将其存入 MATLAB 工作空间，存储时应该给出变量名。

模糊推理问题还可以用 MATLAB 函数 `evalfis()` 求解。该函数的调用格式为 $\boldsymbol{y}$=**evalfis(**$\boldsymbol{X}$**,fis)**，其中，$\boldsymbol{X}$ 为矩阵，其各列为各个输入信号的精确值，`evalfis()` 函数由用户定义的模糊推理系统 `fis` 对输入信号进行模糊化，用该系统进行模糊推理，并将结果进行解模糊化，得出相应的精确输出信号 $\boldsymbol{y}$。

例 7-8 假设已经按上述方式建立起了模糊推理模型，在 x-y 平面内的 $(-3,-5) \sim (3,5)$ 区域内进行网格分割，试用此模糊推理系统绘制出输出的三维曲面。

求解 采用下面的语句可以先读入前面建立的模糊推理系统，并对感兴趣的 x-y 平面区域进行网格分割，将网格数据转换成列向量，再由 `evalfis()` 函数求出曲面的 z 坐标值，这样就可以用下面的语句绘制出三维曲面，如图 7-12 所示。

```
>> fff=readfis('c7mfis.fis');    % 读入模糊推理系统文件
   [x,y]=meshgrid(-3:.2:3,-5:.2:5);   % 进行网格分割
   x1=x(:); y1=y(:); z1=evalfis([x1 y1],fff);   % 模糊推理
   z=reshape(z1,size(x)); surf(x,y,z)    % 绘制曲面
```

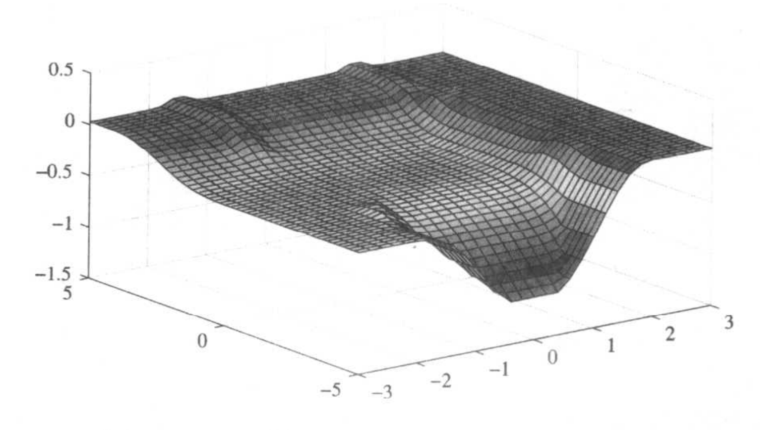


图 7-12 由模糊推理得出的输出曲面

7.1.6 模糊控制系统的仿真方法

利用反馈系统中的误差信号 $e(t)$ 及其变化率 $\mathrm{d}e(t)/\mathrm{d}t$ 来计算控制量的方法称为 PD 控制。典型的模糊 PD 控制器结构框图如图 7-13 所示，其中需要事先引入增益 K_p 和 K_d 分别对误差信号及其变化率信号进行规范处理，使得其值域范围与模糊变量的论域吻合，然后对这两个信号模糊化后得出的信号 (E, E_d) 进行模糊推理，并将得出的模糊量解模糊化，得出精确变量 U，通过规范化增益 K_u 后就可以得出控制信号 $u(t)$。

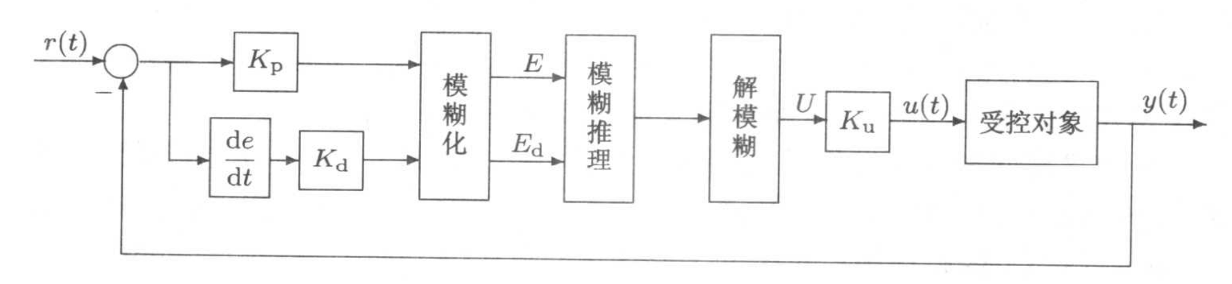


图 7-13 .模糊 PD 控制器控制框图

文献 [7] 采用了更合理的 8 段模糊子集的定义，其示意图如图 7-14 所示。和 7 段模糊子集方式相比，这样的定义将 ZE 集合进一步细化为 NZ (负零) 和 PZ (正零) 两个子集，能更好地刻画在 0 附近误差及其变化率的情况。

从系统的响应看，如果误差 $e(t) = r(t) - y(t)$ 为 PB，则需要给出正的控制量 $u(t)$。进一步地，如果 $\mathrm{d}e(t)/\mathrm{d}t$ 为 NB 和 NM，由于误差大且误差仍有加大的趋势，所以应该加大控制量 $u(t)$，亦即将 $u(t)$ 设置为 PB；相反地，如果误差变化率为 NS 和 NZ，则说明误差有减小的趋势，故无需加大控制量，将其设置为 PM 即可；若变化率为 PZ 或 PS，则应该加更小的控制量，如选择 PS；如果误差变化率为 PM 或 PB，则说明无需加控制量即可消除误差，这时应该选择 $u(t)$ 为 NZ。对其他的 $e(t)$ 与 $\mathrm{d}e(t)/\mathrm{d}t$ 组合当然也可以总结出类似的规则，这样可以得出表

7-2 中给出的各种规则[7]，注意，因为这里的误差定义与该文献的定义差一个符号，故将整个表取了反号。

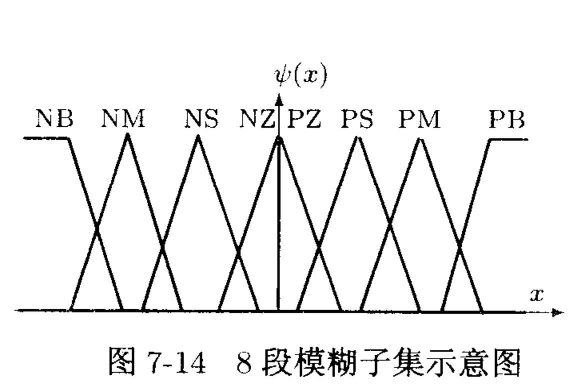


图 7-14 8 段模糊子集示意图

表 7-2 PD 控制器模糊逻辑

		de(t)/dt							
		NB	NM	NS	NZ	PZ	PS	PM	PB
$e(t)$	NB	NB	NB	NM	NM	NS	NS	NZ	NZ
	NM	NB	NB	NM	NM	NS	NS	NZ	NZ
	NS	NB	NB	NM	NS	NS	NZ	NZ	NZ
	NZ	NB	NM	NM	NZ	NS	NZ	PM	PM
	PZ	NM	NM	PZ	PS	PZ	PM	PM	PB
	PS	PZ	PZ	PZ	PS	PS	PM	PB	PB
	PM	PZ	PZ	PS	PS	PM	PM	PB	PB
	PB	PZ	PZ	PS	PS	PM	PM	PB	PB

有了模糊隶属度函数与模糊推理表格，则可以用下面的步骤建立起所需的模糊推理系统模型：

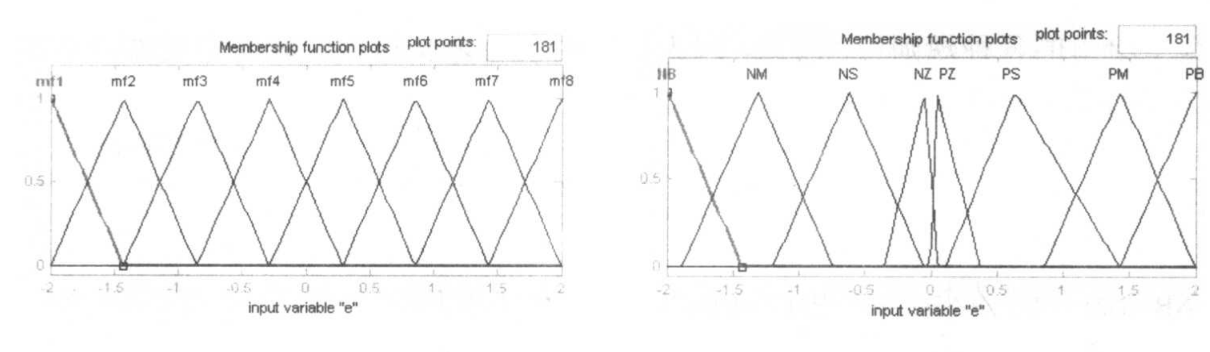
① **启动界面** 在命令窗口中输入 `fuzzy` 命令启动如图 7-7 所示系统界面。

② **信号设定** 在该界面中，默认的系统是单输入单输出的，而建立本模糊推理模型需要双路输入、单路输出，所以应该添加一路输入信号，这可以由菜单项 Edit → Add Variable → Input 添加。分别在图 7-7 所示的界面上修改这三路信号的变量名为 e, ed 和 u。

③ **隶属度函数设置** 双击界面上的输入段 e 图标，将在得出的界面上显示默认的三段模糊子集及隶属度函数曲线。单击 Edit 菜单，其内容如图 7-6 (a) 所示。选择其中的 Remove All MFs 菜单删除默认的所有隶属度函数。修改界面中 Range 栏目中的内容为区间 $[-2,2]$。

选择 Edit → Add MFs 菜单，则可以得出如图所示的对话框，用来输入隶属度函数的模板，对本例问题可以将 Number of MFs 栏目的数值填写为 8，则可以得出默认的 8 段三角形隶属度函数的默认设置，如图 7-15 (a) 所示。将各段隶属度函数的名称依次改成 NB, NM, ⋯，并微调默认隶属度函数的形状，则可以得出如图 7-15 (b) 所示的隶属度函数曲线。用同样的方法对各路输入、输出信号均作同样处理。

④ **编辑模糊推理系统** 选择 Edit → Rules 菜单项，则可以得出如图 7-16 所示的模糊规则编辑界面，在其中将规则逐一输入进该界面。可以由 Add rule 添加规则，用 Change rule 修改规则。对表 7-2 中给出的模糊规则，共需要编辑 64 条规则。



(a) 默认隶属度函数曲线　　　(b) 编辑后的隶属度函数

图 7-15　隶属度函数的编辑

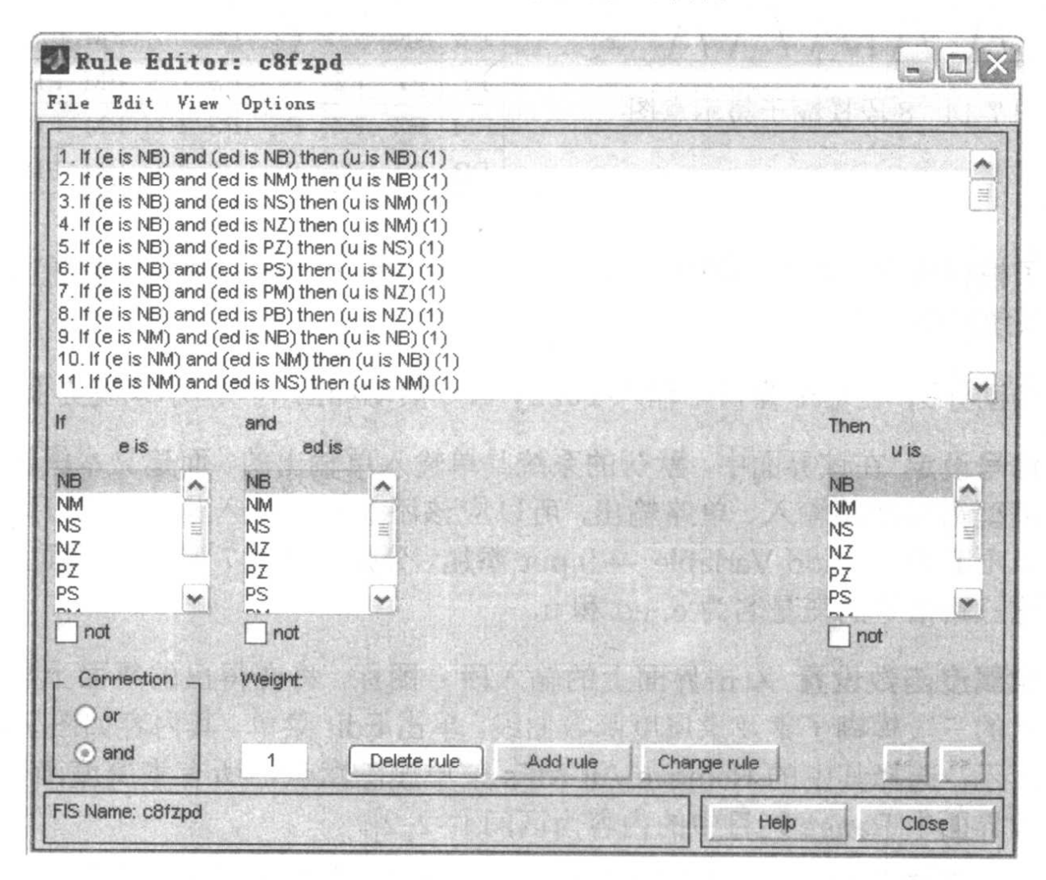
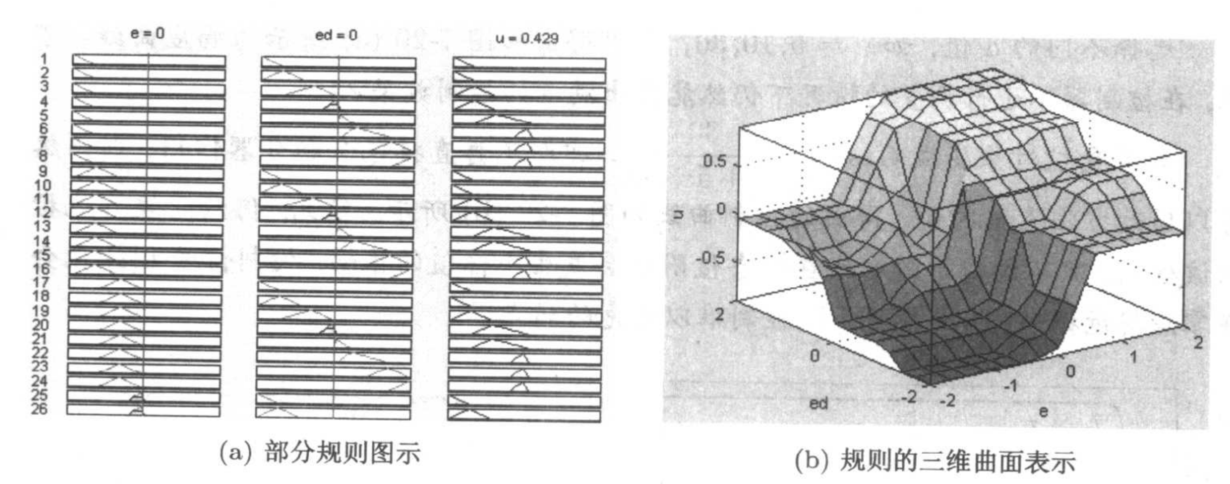


图 7-16　模糊推理规则编辑界面

建立起模糊推理规则后，由 View → Rules 和 View → Surface 菜单项将分别得出如图 7-17 (a)、(b) 所示规则显示图形，由这些图形可以更好地理解建立的模糊推理规则。

⑤ **模糊推理系统的存储**　选择 File → Export 菜单项就可以分别将建立起来的模糊推理系统存成 *.fis 文件或存成 MATLAB 工作空间中的变量。采用这里给出的存储方法，可以将建立起来的模型存储为 c7fzpd.fis。

例 7-9　假设受控对象模型为 $G(s)=\dfrac{30}{s^2+as}$，其中 $a\in[5,50]$，取 $K_{\rm p}=2$，$K_{\rm d}=$



(a) 部分规则图示　　(b) 规则的三维曲面表示

图 7-17　模糊推理规则的图形显示

$K_{\mathrm{u}}=1$，则可以建立起如图 7-18 所示的仿真模型。这里，为了显示其他信号，多设置了 3 个示波器。可以给出如下的命令来对模型进行初始化。

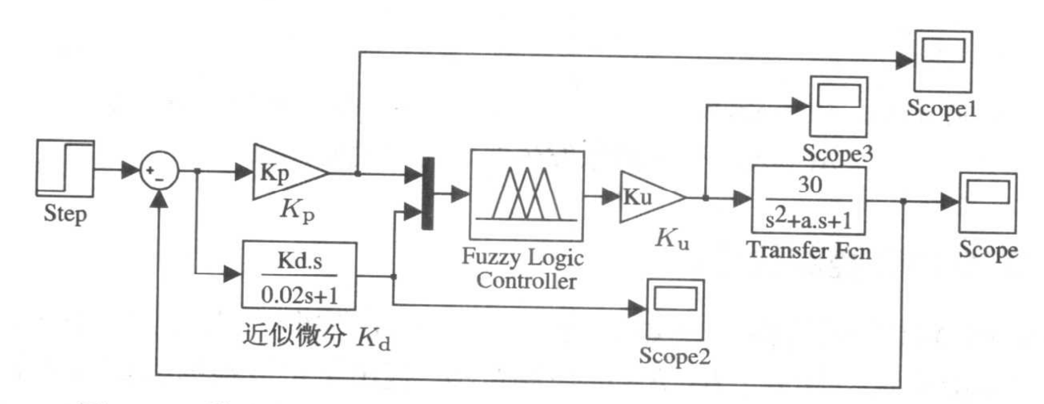
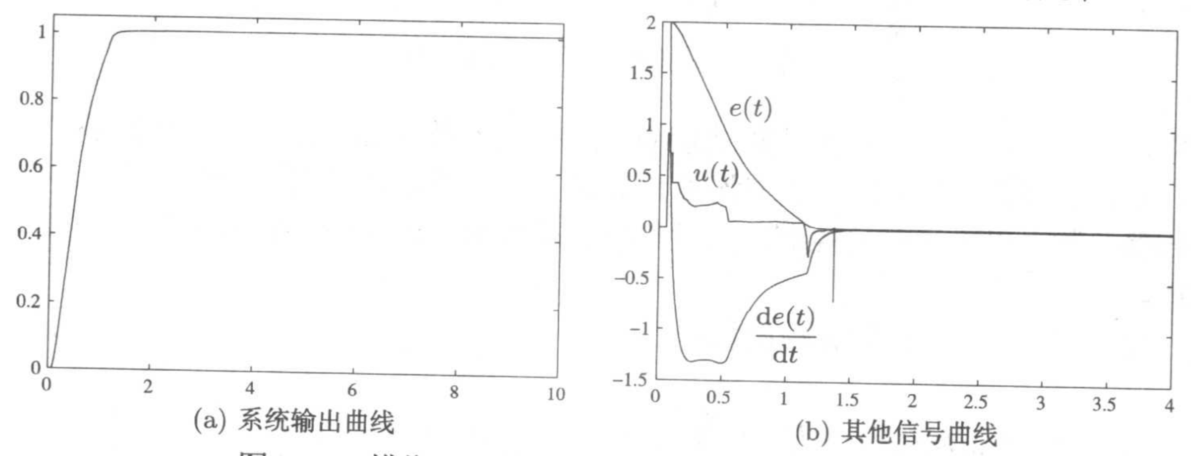


图 7-18　模糊 PD 控制系统的仿真模型 (文件名：c7mfzpd.mdl)

```
>> fuz=readfis('c7fzpd.fis'); a=5; Kp=2; Kd=1; Ku=1;
```

对得出的模型进行仿真，可以得出输出信号，如图 7-19 (a) 所示。其他信号的时域响应曲线如图 7-19 (a) 所示。可见，采用模糊控制可以得出较好的控制效果。

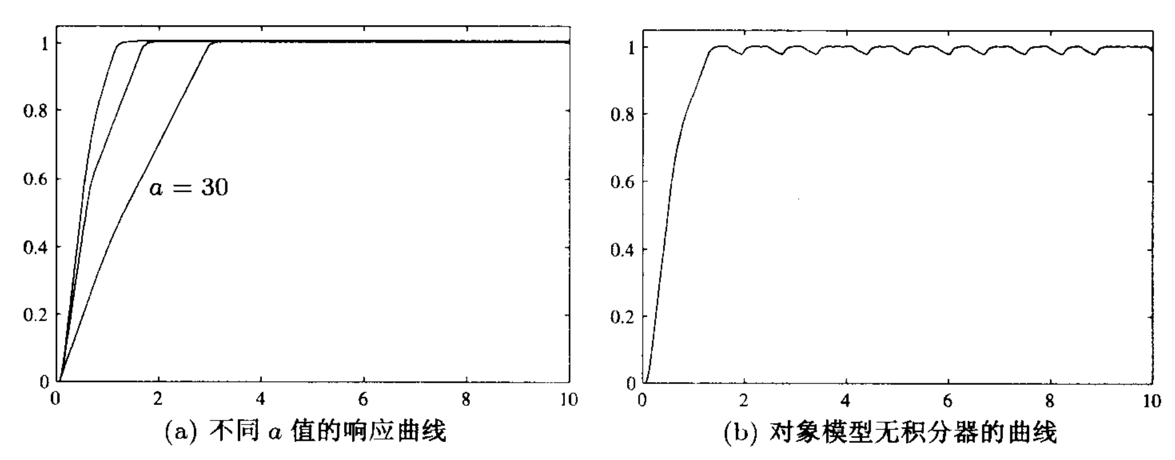


(a) 系统输出曲线　　(b) 其他信号曲线

图 7-19　模糊 PD 控制的输出曲线及其他相关曲线

选择不同的 a 值，如 $a = 5, 10, 30$，可以得出如图 7-20 (a) 所示的响应曲线。可见，在控制器不进行调整的情况下仍然能得出满意的控制效果。

如果受控对象变成 $G(s) = \dfrac{30}{s^2 + 5s + 1}$，亦即不再直接包含积分器作用，则仍然可以用此方法直接控制，得出的控制曲线如图 7-20 (b) 所示。可见，仍然能较好地控制该模型，只不过得出的曲线在稳态值附近会发生小幅值的振动，控制信号 $u(t)$ 也会在零点附近振动，这是模糊 PD 控制难以避免的弱点。



(a) 不同 a 值的响应曲线　　(b) 对象模型无积分器的曲线

图 7-20　受控对象发生变化时的控制效果

7.1.7　模糊 PID 控制器设计

模糊 PID 控制器结构是一类被广泛应用的 PID 控制器，该控制器一改传统 PID 控制器的固定参数 $K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d}$ 的控制策略，提出了可以根据跟踪误差信号等动态改变 PID 控制器参数的方法，以达到改善控制效果，扩大应用范围的目的。

由模糊逻辑整定 PID 控制器的表达式为

$$\begin{cases} K_\mathrm{p}(k) = K_\mathrm{p}(k-1) + \gamma_\mathrm{p}(k)\Delta K_\mathrm{p} \\ K_\mathrm{i}(k) = K_\mathrm{i}(k-1) + \gamma_\mathrm{i}(k)\Delta K_\mathrm{i} \\ K_\mathrm{d}(k) = K_\mathrm{d}(k-1) + \gamma_\mathrm{d}(k)\Delta K_\mathrm{d} \end{cases} \tag{7-1-1}$$

其中，$\gamma_\mathrm{p}(k), \gamma_\mathrm{i}(k), \gamma_\mathrm{d}(k)$ 为校正速度量，随校正次数增加，它们的值将减小。当然，为简单起见，也可以将它们均设置成常数。由整定公式可以看出，下一步的控制器参数可以由当前的控制器参数与模糊推理得出的控制器参数增量的加权和构成。

这时，可以按下式计算控制量

$$u(k) = K_\mathrm{p}(k)e(k) + K_\mathrm{i}(k)\sum_{i=0}^{k} e(i) + K_\mathrm{d}(k)\Big[e(k) - e(k-1)\Big] \tag{7-1-2}$$

注意，这里的求和式子并不是 PID 控制器积分项的全部，正常应该乘以采样周期 T，这里为简单起见，将其含于变量 $K_\mathrm{i}(k)$ 中，上式同样对 $K_\mathrm{d}(k)$ 进行了相应处理。由于计算 $\sum_{i=0}^{k} e(i)$ 较困难，所以应该引入状态变量 $x(k)=\sum_{i=0}^{k} e(i)$。这样可以推导出状态方程为

$$x(k+1)=x(k)+e(k) \tag{7-1-3}$$

这时，式 (7-1-2) 中控制量可以改写成

$$u(k)=K_\mathrm{p}(k)e(k)+K_\mathrm{i}(k)x(k)+K_\mathrm{d}(k)\Big[e(k)-e(k-1)\Big] \tag{7-1-4}$$

模糊 PID 控制器的典型结构如图 7-21 所示。由于直接用模块搭建前面的模糊 PID 控制器算法比较复杂，所以这里采用 S-函数的形式来构造该模块。分析前面介绍的算法，可见状态变量个数为 1；输出个数可以选择为 1 个，但考虑到本例子还想显示变化的 $K_\mathrm{p},K_\mathrm{i},K_\mathrm{d}$ 系数，所以暂时选择输出个数为 4；输入信号可以选择两路，即 $\boldsymbol{u}(k)=[e(k),e(k-1)]^\mathrm{T}$。这样可以容易地编写出如下的 S-函数来表示模糊 PID 控制器的核心部分。

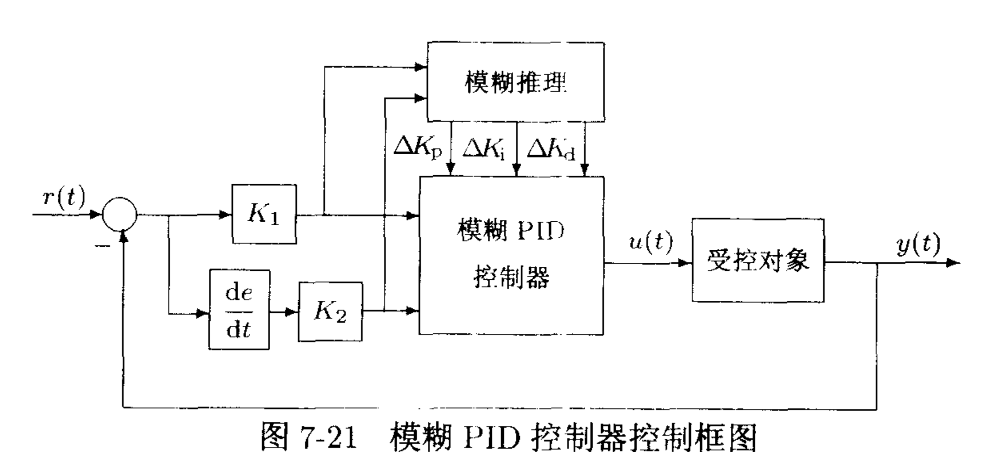
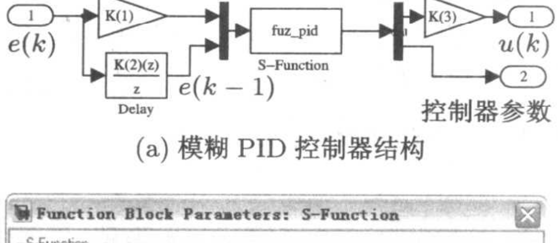


图 7-21 模糊 PID 控制器控制框图

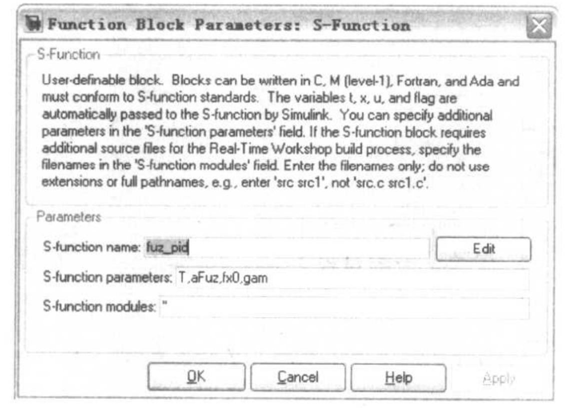
```
function [sys,x0,str,ts]=fuz_pid(t,x,u,flag,T,aFuz,fx0,gam)
switch flag,
   case 0, [sys,x0,str,ts] = mdlInitializeSizes(T);
   case 2, sys = mdlUpdates(x,u);
   case 3, sys = mdlOutputs(x,u,T,aFuz,fx0,gam);
   case {1, 4, 9}, sys = [];
   otherwise, error(['Unhandled flag = ',num2str(flag)]);
end;
% --- 模块初始化函数 mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes(T)
sizes = simsizes; % 读入系统变量的默认值
sizes.NumContStates = 0; sizes.NumDiscStates = 3;
sizes.NumOutputs = 4; sizes.NumInputs = 2;
```

```
sizes.DirFeedthrough = 0; sizes.NumSampleTimes = 1;
sys = simsizes(sizes); x0 = zeros(3,1);
str = []; ts = [T 0]; % 继承输入信号的采样周期
% --- 离散状态更新函数 mdlUpdate
function sys = mdlUpdates(x,u)
sys=[u(1);  x(2)+u(1); u(1)-u(2)];  % PID
% --- 输出量计算函数 mdlOutputs
function sys = mdlOutputs(x,u,T,aFuz,fx0,gam)
Kpid=fx0+gam(:).*evalfis(x([1,3]),aFuz)'; sys=[Kpid'*x; Kpid];
```

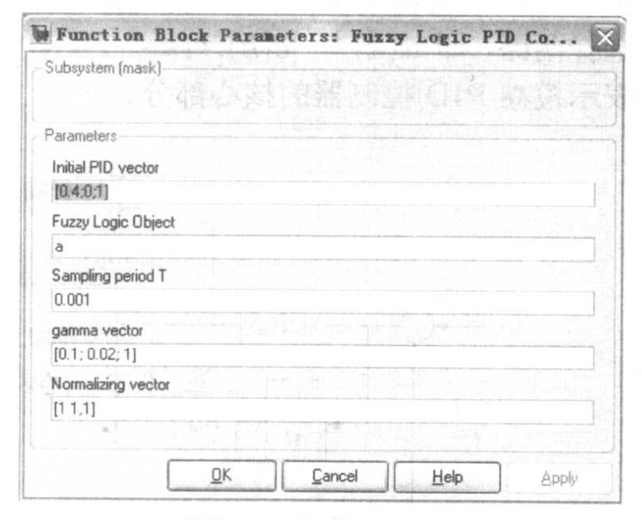
有了核心的 S-函数，则可以构造并封装出模糊 PID 控制模块，其内部结构如图 7-22 (a) 所示。该图中引用了 S-函数 fuz_pid.m，其参数对话框如图 7-22 (b) 所示，整个 PID 控制器的参数设置对话框如图 7-22 (c) 所示。



(a) 模糊 PID 控制器结构



(b) fuz_pid 模块参数设置



(c) 模糊 PID 控制器参数对话框

图 7-22　模糊 PID 控制器模块设计

在模糊 PID 控制器中，根据经验可以构造出表 7-3 中给出的参数变化表[8]，根据该模糊表可以在 MATLAB 环境下输入该模糊推理系统，该系统仍有两个输入，但和前面不同的是，该系统将有三路输出，分别对应于 $\Delta K_{\rm p}$, $\Delta K_{\rm i}$ 和 $\Delta K_{\rm d}$。

根据模糊规则表，可以用 `fuzzy()` 函数可视地建立起整个模糊推理系统 c7fuzpid.fis，该系统有两路输入和三路输出，如图 7-23 所示。该模型中选择输入和输出变量的范围均为 $(-3,3)$，为方便起见，应该保持该模糊推理系统的输入、输出变量范围，而推理结果可以由系数 $(K_1,K_2,\gamma_{\rm p},\gamma_{\rm i},\gamma_{\rm d},K_{\rm u})$ 来修正。

在模糊系统中，模糊规则编辑程序界面如图 7-24 所示，而得出的 3 个规则曲面在图 7-25 中给出。读者若想了解该模糊推理系统的具体内容和参数，可以用 `fuzzy()` 界面打开 c7fuzpid.fis 文件。

表 7-3 PID 控制器模糊逻辑

		de(t)/dt																				
		ΔK_p							ΔK_i							ΔK_d						
		NB	NM	NS	ZE	PS	PM	PB	NB	NM	NS	ZE	PS	PM	PB	NB	NM	NS	ZE	PS	PM	PB
$e(t)$	NB	PB	PB	PM	PM	PS	ZE	ZE	NB	NB	NM	NM	NS	ZE	ZE	PS	NS	NB	NB	NB	NM	PS
	NM	PB	PB	PM	PS	PS	ZE	ZE	NB	NB	NM	NS	NS	ZE	ZE	PS	NS	NB	NB	NB	NM	PS
	NS	PM	PM	PM	PM	ZE	NS	NS	NB	NM	NS	NS	ZE	PS	PS	ZE	NS	NM	NM	NS	NS	ZE
	ZE	PM	PM	PS	ZE	NS	NM	NM	NM	NM	NS	ZE	PS	PM	PM	ZE	NS	NS	NS	NS	NS	ZE
	PS	PS	PS	ZE	NS	NS	NM	NM	NM	NS	ZE	PS	PS	PM	PB	ZE	ZE	ZE	ZE	ZE	ZE	ZE
	PM	PS	ZE	NS	NM	NM	NM	NB	ZE	ZE	PS	PS	PM	PB	PB	PB	NS	PS	PS	PS	PS	PB
	PB	ZE	ZE	NM	NM	NM	NB	NB	ZE	ZE	PS	PM	PM	PB	PB	PB	PM	PM	PM	PS	PS	PB

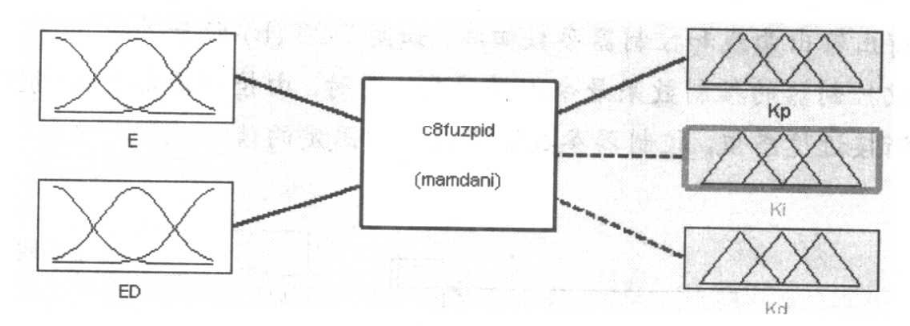


图 7-23 模糊推理系统结构图

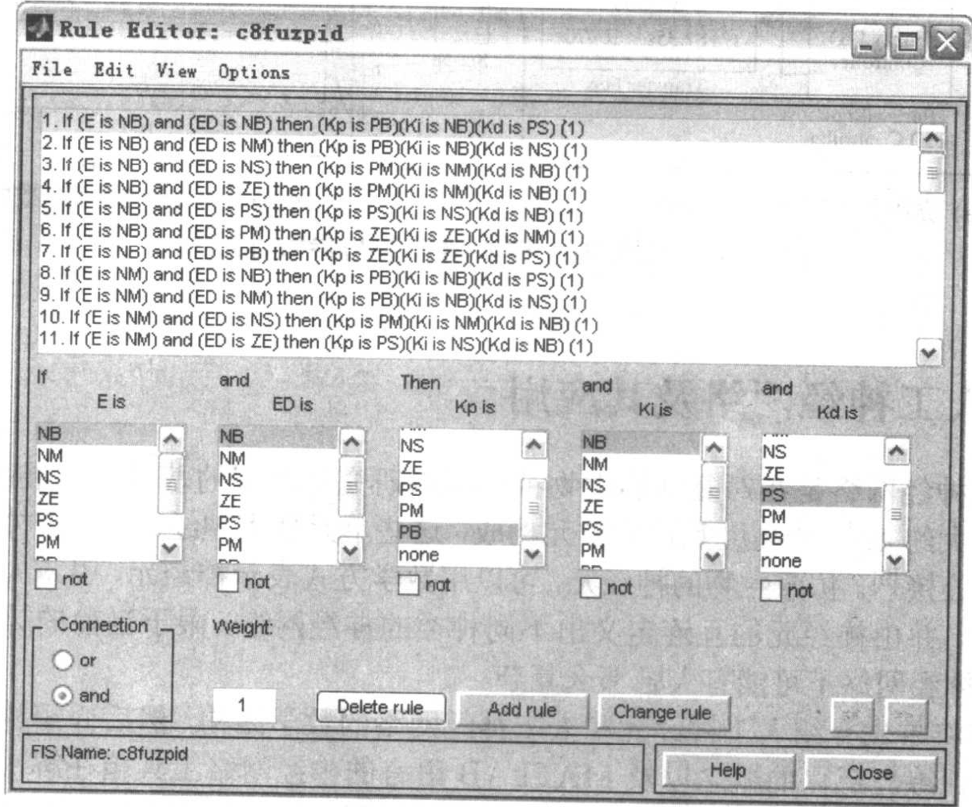
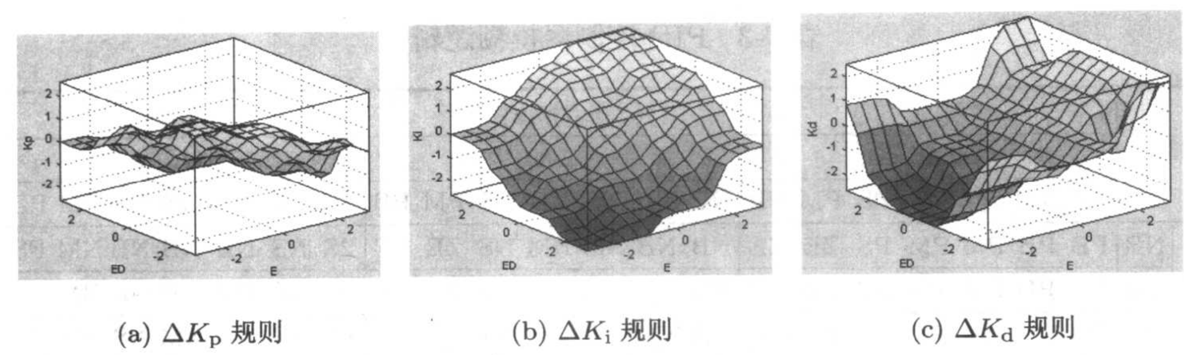


图 7-24 模糊规则编辑对话框

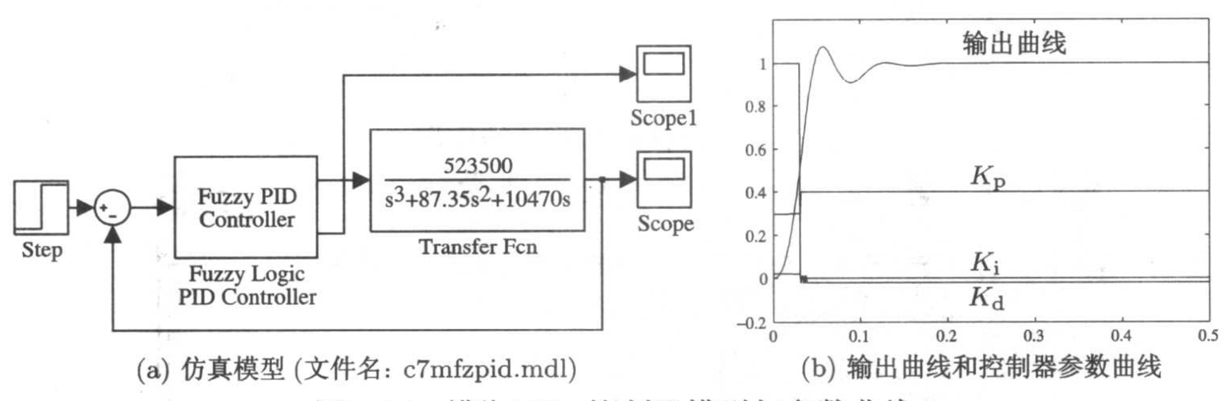


(a) ΔK_p 规则　　(b) ΔK_i 规则　　(c) ΔK_d 规则

图 7-25　模糊 PID 控制器三参数的模糊推理规则曲面

例 7-10 假设受控对象为 $G(s)=\dfrac{523500}{s^3+87.35s^2+10470s}$，选择 $K_1=K_2=K_\mathrm{u}=1$ 及 $\gamma=[0.1,0.02,1]^\mathrm{T}$，这样可以建立起如图 7-26 (a) 所示的仿真框图，对该系统进行仿真则可以得出输出曲线和控制器参数曲线，如图 7-26 (b) 所示。

可见，该控制器的控制效果是令人满意的，同时，由控制器参数曲线显示出随着系统输出逐渐接近稳态值，控制器参数逐渐稳定到固定的值。



(a) 仿真模型 (文件名：c7mfzpid.mdl)　　(b) 输出曲线和控制器参数曲线

图 7-26　模糊 PID 控制器模型与参数曲线

7.2　人工神经网络及其应用

人工神经网络是在对复杂的生物神经网络研究和理解的基础上发展起来的。人脑是由大约 10^{11} 个高度互连的单元构成，这些单元称为神经元，每个神经元约有 10^4 个连接 [9]。仿照生物的神经元，可以用数学方式表示神经元，引入人工神经元的概念，并由神经元的互连定义出不同种类的神经网络。限于当前的计算机水平，人工神经网络不可能有人脑那么复杂。

本节将首先介绍人工神经元和人工神经网络的数学结构，然后介绍神经网络的建立、训练与泛化的概念以及 MATLAB 语言的神经网络工具箱在解决这些问题中的应用，最后介绍一个用户界面来利用神经网络求解数据拟合中的问题。

7.2.1 神经网络基础知识

1. 神经网络的概念及结构

单个人工神经元的数学表示形式如图 7-27 所示。其中，$x_1, x_2, \cdots, x_n$ 为一组输入信号，它们经过权值 w_i 加权后求和，再加上阈值 b，则得出 u_i 的值，可以认为该值为输入信号与阈值所构成的广义输入信号的线性组合。该信号经过传输函数 $f(\cdot)$ 可以得出神经元的输出信号 y。

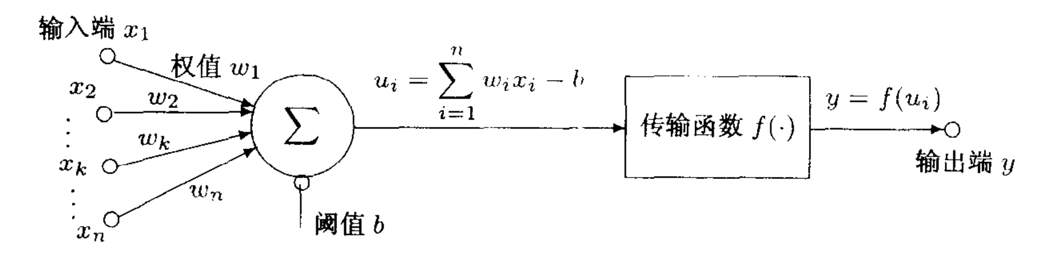


图 7-27 人工神经元的基本结构

在人工神经元中，权值和传输函数是两个关键的因素。权值的物理意义是输入信号的强度，若涉及多个神经元则可以理解成神经元之间的连接强度。神经元的权值 w_i 应该通过神经元对样本点反复的学习过程而确定，而这样的学习过程在神经网络理论中又称为训练。传输函数又称为激励函数，可以理解成对 u_i 信号的非线性映射，一般的传输函数应该为单值函数，使得神经元是可逆的。常用的传输函数有 Sigmoid 函数和对数 Sigmoid 函数，它们的数学表达式分别为

$$\begin{aligned}&\text{Sigmoid 函数}\ \ f(x)=\frac{2}{1+\mathrm{e}^{-2x}}-1=\frac{1-\mathrm{e}^{-2x}}{1+\mathrm{e}^{-2x}}\\&\text{对数 Sigmoid 函数}\ \ f(x)=\frac{1}{1+\mathrm{e}^{-x}}\end{aligned}\tag{7-2-1}$$

当然也可以使用简单的饱和函数和阶跃函数作为传输函数。下面将通过例子介绍各类传输函数的形状及基于 MATLAB 神经网络工具箱的绘制方法。

例 7-11 试绘制出各种常用的传输函数曲线。

求解 用下面的语句可以直接绘制出 Sigmoid 函数的曲线，如图 7-28 所示。

```
>> x=-2:0.01:2; y=tansig(x); plot(x,y)
```

用 `logsig()` 语句取代前面的 `tansig()` 函数则得出对数 Sigmoid 函数曲线。另外，由其他函数可以绘制出另外几种传输函数曲线，如图 7-28 所示，同时给出了绘制这些传输函数的 MATLAB 函数名。

由若干个神经元相互连接，则可以构成一种网络，称为神经网络。由于连接方式的不同，神经网络的类型也将不同。这里仅介绍前馈神经网络，因为其权值训练中采用误差逆向传播的方式，所以这类神经网络更多地称为反向传播 (back

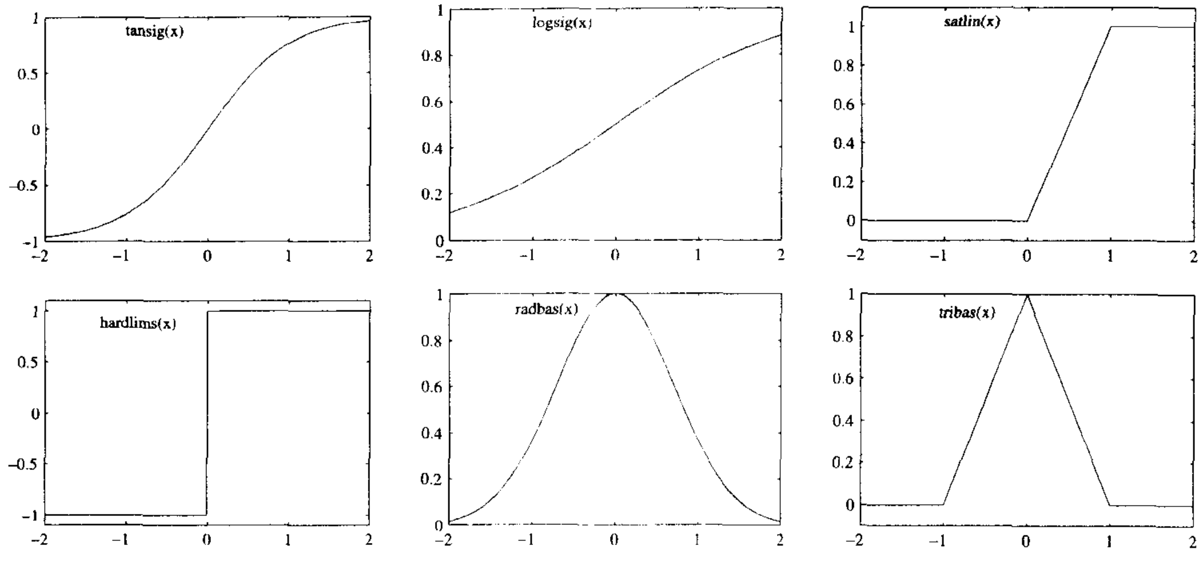


图 7-28 各种传输函数曲线

propagation) 神经网络，简称 BP 网。BP 网的基本网络结构如图 7-29 所示。在 MATLAB 神经网络工具箱中认为这样网络的层数为 $k+1$，其中前 k 层为隐层，第 $k+1$ 层为输出层，其节点个数为 m。

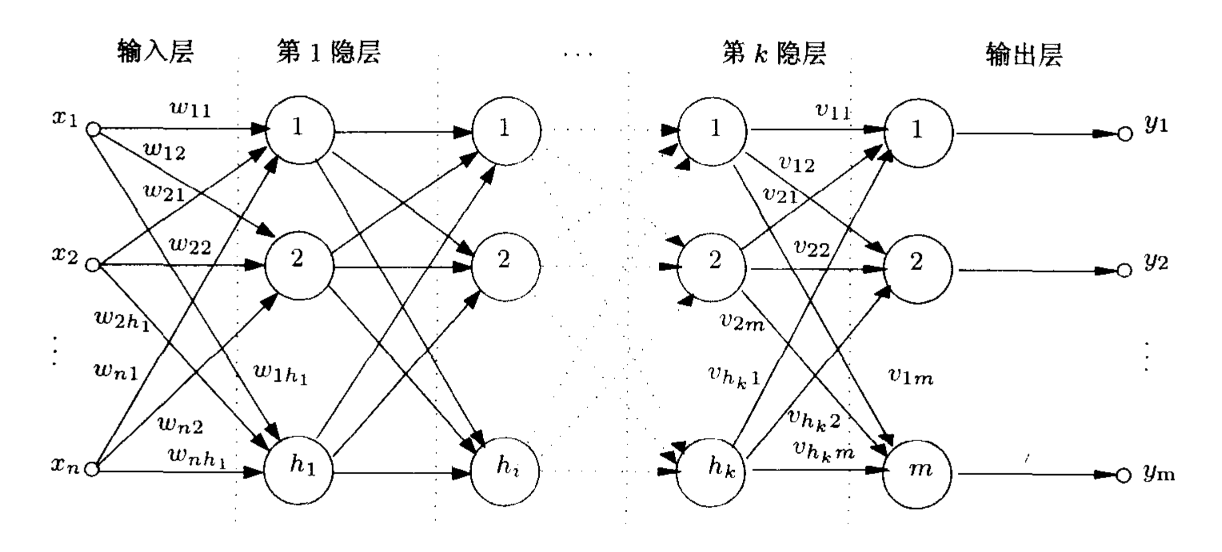


图 7-29 神经元的基本结构

利用 MATLAB 语言的神经网络工具箱提供的现成函数和神经网络类，建立起一个前馈的 BP 神经网络模型还是很容易的。可以使用 `newff()` 函数，具体的语句格式为

$$\texttt{net=newff}([\boldsymbol{x}_\mathrm{m},\boldsymbol{x}_\mathrm{M}],[h_1,h_2,\cdots,h_k],\{f_1,f_2,\cdots,f_k\})$$

其中，$\boldsymbol{x}_\mathrm{m}$ 和 $\boldsymbol{x}_\mathrm{M}$ 分别为列向量，存储各个样本输入数据的最小值和最大值。若样本数据已知，还可以用 `min()` 和 `max()` 函数求出，第 2 个输入变量是一个行向

量，用户可以将神经网络各层的节点数输入，单元的个数为隐层层数。函数的第 3 个输入变量为单元式数组，由若干个字符串构成，每个字符串对应于该层的传输函数类型。注意，这里要求同一层应该使用相同的传输函数。给了这些信息后，就可以构造出神经网络数据对象 net，该对象的一些重要属性在表 7-4 中给出。下面通过例子演示神经网络对象的建立。

表 7-4　神经网络对象的常用属性

属性名	数据类型	属性说明	默认参数
net.IW	单元数组	输入层和隐层加权，其中 net.IW{1} 为输入层的加权矩阵，net.IW{$i+1$} 为第 i 隐层的加权矩阵	随机
net.numInputs	整型	输入路数，可以由 x_m 或 x_M 维数自动计算	
net.numLayers	整型	隐层数，可以由 newff() 语句的调用参数确定	
net.LW	单元数组	输入层和隐层加权，其中 net.IW{1} 为输入层的加权矩阵，net.IW{$i+1$} 为第 i 隐层的加权矩阵	随机
net.trainParam.epochs	整型	最大训练步数，当误差准则满足，即使未训练到此步骤也将停止训练，返回训练结果	100
net.trainParam.lr	实型	自学习的学习率	0.01
net.trainParam.goal	实型	训练误差准则，当误差小于此值时停止训练	0
net.trainFcn	字符串	训练算法，可选择 'traincgf'（共轭梯度法）、'train'（批处理训练算法）、'traingdm'（带动量的梯度下降算法）、'trainlm'（Levenberg-Marquardt 算法）等	'train'

例 7-12 假设输入信号为 2 路，其信号范围分别为 $[0,1]$ 和 $(-1,5)$，且输出信号为单路信号，试利用 newff() 函数建立所需的前馈网络对象。

求解 首先考虑建立一个前馈网络，使其有 2 个隐层，其中第 1 隐层有 8 个节点，且该层神经元均采用对数型 Sigmoid 传输函数，第 2 层的节点个数应该等于输出信号的路数，故只能选择其节点数为 1，并假设该层采用的传输函数为 Sigmoid 函数。这样可以用下面的语句建立所需的前馈神经网络模型。

```
>> net=newff([0,1; -1,5],[8,1],{'tansig','logsig'});
```

若需要建立含有 3 个隐层的神经网络，令第 1 层有 4 个节点，传输函数为线性函数，第 2 层有 6 个节点，传输函数为 logsig()，第 3 层有一个节点，使用的传输函数为 tansig()，这样就可以建立起所需的神经网络模型。

```
>> net=newff([0,1; -1,5],[4 6 1],{'purelin','tansig','logsig'});
```

除了神经网络结构之外，还可以用下面的语句格式设定其他参数，如

```
net.trainParam.epochs=300,  net.trainFcn='trainlm'
```

2. 神经网络的训练与泛化

若建立了神经网络模型 net，则可以调用 train() 函数对神经网络参数进行训练。该函数的调用格式为 [net,tr,$\boldsymbol{Y}_1$,$\boldsymbol{E}$]=train(net,$\boldsymbol{X}$,$\boldsymbol{Y}$)，其中，变量 $\boldsymbol{X}$ 为 $n\times M$ 矩阵，n 为输入变量的路数，M 为样本的组数；$\boldsymbol{Y}$ 为 $m\times M$ 矩阵，m 为输出变量的路数；$\boldsymbol{X},\boldsymbol{Y}$ 分别存储样本点的输入和输出数据。由样本点数据进行训练，则可以得出训练后的神经网络对象 net，且可以返回其他相关的内容，tr 为结构体数据，返回训练的相关跟踪信息，tr.epochs 为训练步数，tr.perf 为各步目标函数的值。$\boldsymbol{Y}_1$ 和 $\boldsymbol{E}$ 矩阵分别返回由神经网络计算出的输出和误差矩阵。在训练过程中将每隔 25 步自动显示一次训练指标。训练结束后还可以用下面的语句绘制出目标值曲线 plotperf(tr) 。

如果在给出的最大训练步数下无法得出满足要求的网络，则将给出错误的信息提示。用户可以再调用该函数一次，这时将以上次的训练结果加权矩阵为初值继续训练，用户可以循环调用该语句。如果误差在几次循环后仍无显著改善，则说明网络结构有问题，应该修改网络结构。

神经网络训练完成后，可以利用该网络对样本区域内的其他输入量求解其输出值，这种求值的方法称为神经网络的仿真或泛化 (generalization)，可以理解为利用神经网络进行数据拟合，对新的输入点数据 $\boldsymbol{X}_1$ 调用 sim() 函数进行泛化，得出这些输入点处的输出矩阵 $\boldsymbol{Y}_1$，且 $\boldsymbol{Y}_1$=sim(net,$\boldsymbol{X}_1$) 。

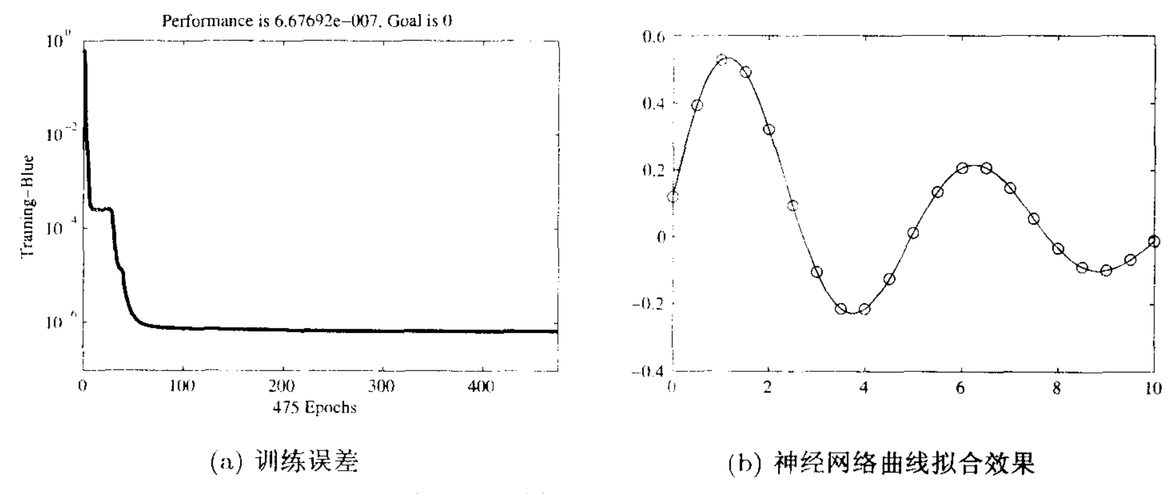
神经网络是否成功不在于对样本点本身拟合误差的大小，而关键在于其泛化效果。如果对样本点以外的其他输入点均有较好的拟合，则说明该神经网络结构合理。否则，训练出来的神经网络没有应用价值。下面将通过例子来演示神经网络及其在数据拟合中的应用以及神经网络控制参数对训练的影响。

例 7-13 由已知函数 $y(x)=0.12\mathrm{e}^{-0.213x}+0.54\mathrm{e}^{-0.17x}\sin(1.23x)$ 生成一组数据，试用神经网络对其进行拟合。

求解 可以用下面的语句输入样本点数据，并选择前馈神经网络。设有 2 个隐层，因为最后一个隐层实际上为输出层，所以其节点个数应该与输出路数一致，故节点数为 1。现在令第 1 隐层节点个数为 5，则可以用下面的语句进行神经网络训练，并选择更密集的输入数据进行泛化，得出如图 7-30 (a) 所示的训练误差和如图 7-30 (b) 所示的泛化效果。可见，这样得出的拟合效果是令人满意的，和理论曲线之间看不出任何差异。

```
>> f=@(x)0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
   x=0:.5:10; y=f(x); x0=[0:0.1:10]; y0=f(x0);
   net=newff([0,10],[5,1],{'tansig','tansig'});
   net.trainParam.epochs=1000;   % 设置最大步数
```

```
net=train(net,x,y);     % 训练神经网络
figure; y1=sim(net,x0); plot(x,y,'o',x0,y0,x0,y1,':');
```



(a) 训练误差　　(b) 神经网络曲线拟合效果

图 7-30　神经网络拟合效果

在训练后的模型中，net.IW{1} 和 net.LW{2} 分别为两层的权值

$$\boldsymbol{w}^{\mathrm{T}}=\begin{bmatrix}0.70446 & -0.86985 & 0.51921 & -0.31618 & 0.99491\end{bmatrix}$$

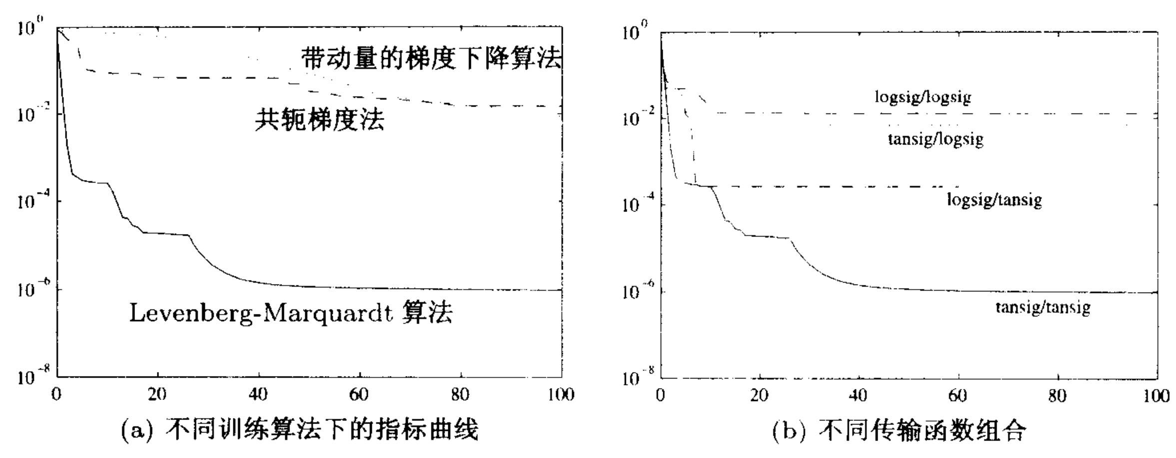
$$\boldsymbol{v}=\begin{bmatrix}-0.3203 & -0.17924 & 1.6316 & 2.6872 & 1.1194\end{bmatrix}$$

选择不同的训练算法，则将得出不同的误差曲线，如图 7-31 (a) 所示。

```
>> net=newff([0,10],[5,1],{'tansig','tansig'});
   net.trainParam.epochs=100;
   net.trainFcn='trainlm'; [net,b1]=train(net,x,y);
   net=newff([0,10],[5,1],{'tansig','tansig'});
   net.trainFcn='traincgf'; [net,b2]=train(net,x,y);
   net=newff([0,10],[5,1],{'tansig','tansig'});
   net.trainFcn='traingdx'; [net,b3]=train(net,x,y);
   semilogy(b1.epoch,b1.perf); hold on
   semilogy(b2.epoch,b2.perf,'--'); semilogy(b3.epoch,b3.perf,':')
```

从训练效果看，用其他算法很多步数难以达到的训练效果用 Levenberg-Marquardt 算法可以较少步数就能得出满意的效果。下面再讨论各层传输函数对训练的影响，可以给出如下命令对不同隐层组合进行试验，得出如图 7-31 (b) 所示的结果。可见，两层均采用 tansig() 函数的训练效果明显好于其他组合，故在实际训练中若没有特殊要求，应该采用 Sigmoid 函数。

```
>> net=newff([0,10],[5,1],{'tansig','logsig'});
   net.trainParam.epochs=100;
   n1.trainFcn='trainlm'; [n1,b2]=train(n1,x,y);
```



(a) 不同训练算法下的指标曲线　　(b) 不同传输函数组合

图 7-31　神经网络控制参数及其作用

```
A=newff([0,10],[5,1],{'logsig','tansig'}); [A,b3]=train(A,x,y);
B=newff([0,10],[5,1],{'logsig','logsig'}); [B,b4]=train(B,x,y);
semilogy(b1.epoch,b1.perf); hold on;
semilogy(b2.epoch,b2.perf,'--');
semilogy(b3.epoch,b3.perf,':'); semilogy(b4.epoch,b4.perf,'-.')
```

若增加隐层节点个数，例如增加到 15，则可以重新建立神经网络，并进行训练。从训练结果看，只用 50 步就能得出极小的拟合误差，如图 7-32 (a) 所示。得出的曲线拟合结果如图 7-32 (b) 所示。

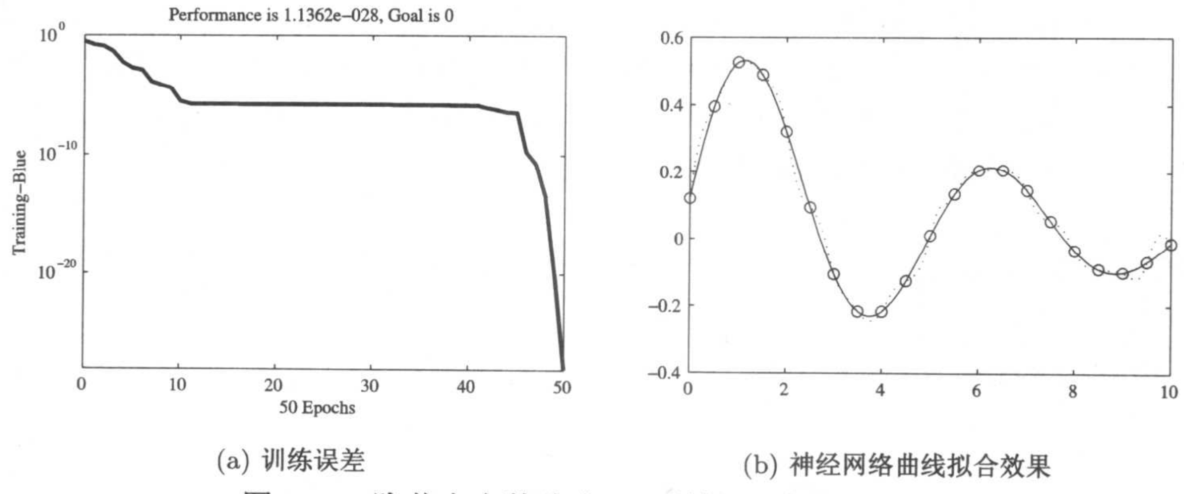
```
>> net=newff([0,10],[15,1],{'tansig','tansig'});
   net.trainParam.epochs=100;
   net.trainFcn='trainlm'; [net,b2]=train(net,x,y);
   figure; y1=sim(net,x0); plot(x0,y0,x0,y1,x,y,'o')
```

从得出的拟合结果看，似乎无限增大隐层节点个数就可以改进拟合效果。其实不然，增大节点个数会改善对样本点的拟合，但对其他点的函数拟合将得出如图 7-32 (b) 所示的结果，亦即神经网络泛化出现了问题，故不能无限增加节点个数。不过节点个数如何选择至今没有公认的解析方法，只能根据实际情况用试凑方式选择。

7.2.2　神经网络界面

MATLAB 的神经网络工具箱提供了一个可以直接使用的程序。在 MATLAB 命令窗口中给出 **nntool** 命令，就可以打开一个如图 7-33 所示的图形用户界面，可以用该界面建立所需的神经网络模型，并由已知数据对该网络进行训练、仿真。下面将通过例子演示神经网络图形界面的使用。

例 7-14　考虑例 7-13 中给出的数据和神经网络拟合效果，试利用神经网络工具箱的 nntool 界面完成同样的拟合。



(a) 训练误差 (b) 神经网络曲线拟合效果

图 7-32 隐节点个数选为 15 时神经网络拟合效果

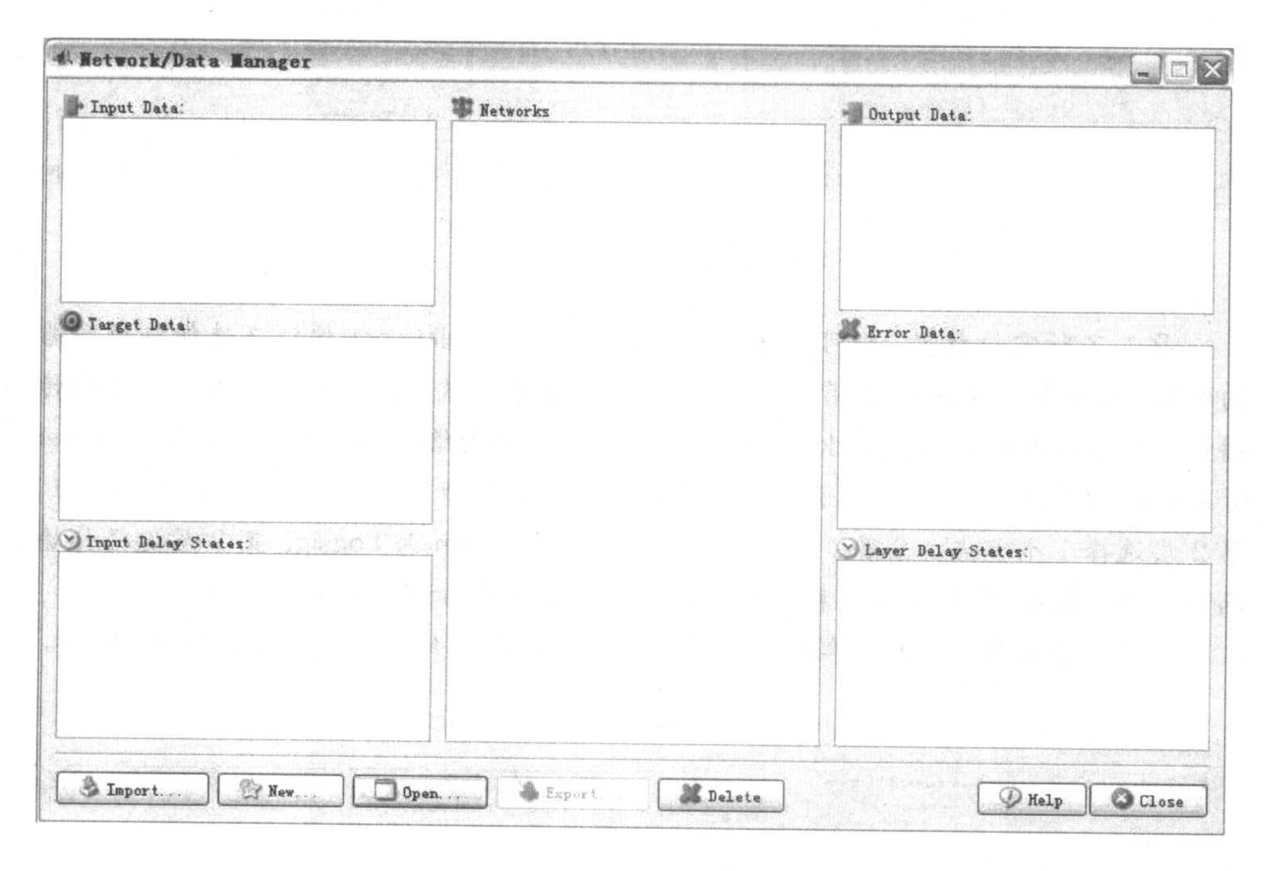


图 7-33 神经网络应用界面

求解 先输入已知数据，然后启动 nntool，得出如图 7-33 所示的程序界面。可以通过这个界面对给定的数据进行神经网络拟合。

```
>> f=@(x)0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
   x=0:.5:10; y=f(x); x0=[0:0.1:10]; y0=f(x0);
   nntool      % 启动神经网络拟合界面
```

如果想利用神经网络拟合系统，首先需要将数据输入到此界面。这可以通过 Import 按钮来实现，单击该按钮将弹出如图 7-34 所示的对话框。将中间栏目下的 x, xx 两个现有的 MATLAB 工作空间变量作为输入变量 (右面栏目的 Inputs) 输入，将变量 y 作为目标变量 (亦即 Targets) 输入到界面中。

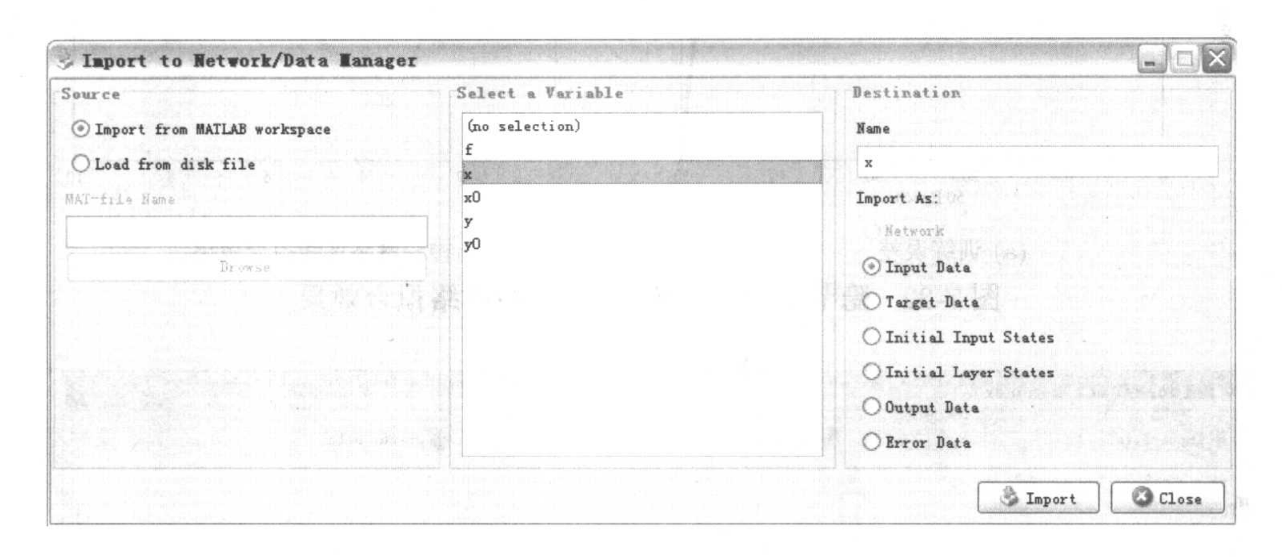
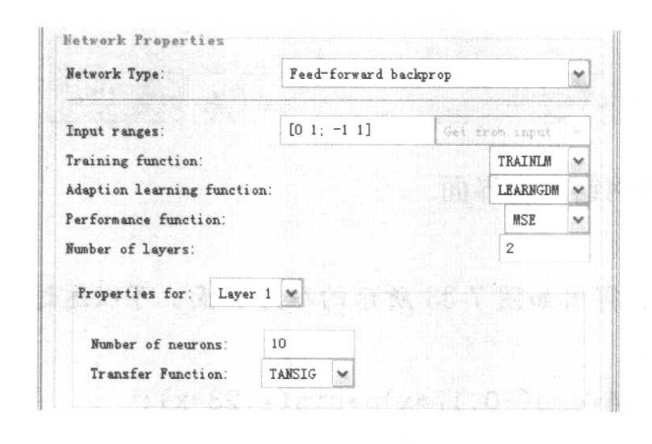


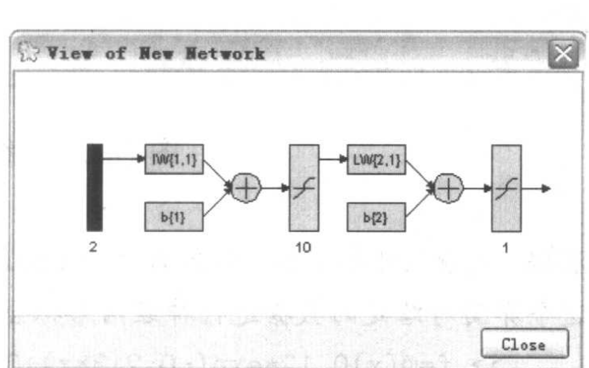
图 7-34　数据输入界面

导入了所需的数据，则可以单击主界面中的 New Network 按钮来选择神经网络的结构，这样将得出如图 7-35 (a) 所示的界面。其中默认的网络结构是前馈型反向传播神经网络 (Feedforward Backprop)。保持各个默认的网络结构，将网络层数 (Number of layers) 设置成 2，在下面的列表框中分别选择不同的节点数，第 1 层选择 8 个节点，第 2 层选择 1 个节点。在第 1 层中选择 Transfer Function 为 Logsig，第 2 层选择传输函数为 Pureline，单击 Create 按钮就可以确定神经网络结构，关闭此窗口。

神经网络结构确定后，单击 View 即可以显示神经网络结构，如图 7-35 (b) 所示。



(a) 神经网络结构设置对话框



(b) 神经网络结构

图 7-35　BP 网络结构设置与显示

现在可以训练神经网络了。单击 Train 按钮，得出如图 7-36 所示的对话框。在对话框中需要输入训练用已知数据，如在 Inputs 和 Targets 栏目分别填写 x 和 y。另外，还可以单击 Training Parameters 标签指定神经网络训练的控制参数，得出如图 7-37 所示的对话框，例如将训练步数 epochs 设置为 1000，这时程序将自动训练网络参数，当误差指标满足时会自动停止训练，得出所需的参数。

图 7-36 神经网络训练参数设置对话框

图 7-37 网络训练控制参数对话框

单击 Train 按钮就可以开始训练，得出如图 7-30 (a) 所示的训练误差曲线。可见，这样的训练误差很小，将训练得出的网络模型输出 (Export) 到 MATLAB 环境中，这时将由下面的语句绘制出曲线拟合与泛化结果，如图 7-30 (b) 所示。

```
>> y1=sim(network1,x0); plot(x,y,'o',x0,y0,x0,y1,':')
```

7.2.3 神经网络控制系统仿真

MATLAB 的神经网络工具箱模块集提供了三个不同的神经网络控制器模型，分别为神经网络预测控制器、神经网络参考模型控制器和反馈线性化神经网络控制器，这些模块可以直接用于控制系统的仿真研究。神经网络控制器模块集如图 7-38 所示。这里只以其中的神经网络预测控制为例介绍基于神经网络的控制及仿真方法。

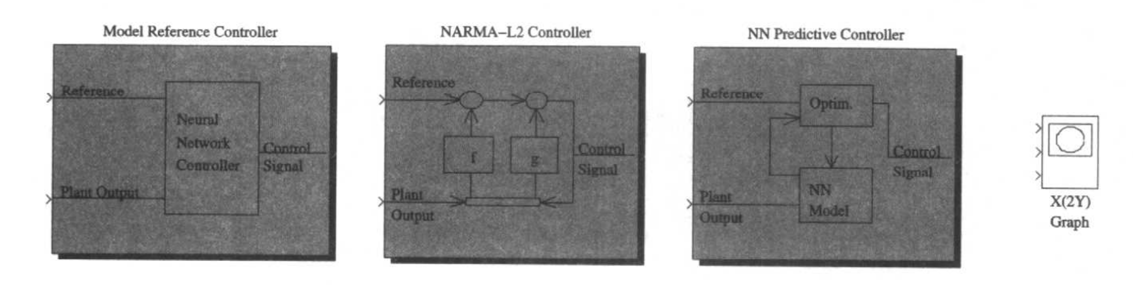


图 7-38 神经网络控制模块集

和以前介绍的模型预测控制类似，所谓基于神经网络的模型预测即采用神经网络去近似受控对象模型，通过必要的训练建立起预测信号 $y_{\rm m}(t+j)$ 的神经网络模型。引入滚动优化的性能指标

$$J=\sum_{j=N_1}^{N_2}[y_{\rm r}(t+j)-y_{\rm m}(t+j)]^2+\rho\sum_{j=1}^{M}\Delta u^2(t+j) \tag{7-2-2}$$

其中 $y_{\rm r}(t+j)$ 为期望的输出信号，(N_1,N_2) 为预测时域，M 为控制时域，ρ 为输入信号变化率的加权。求解相应的最优化问题则可以得出控制信号序列 $u(t)$。基于神经网络的模型预测控制结构如图 7-39 所示。

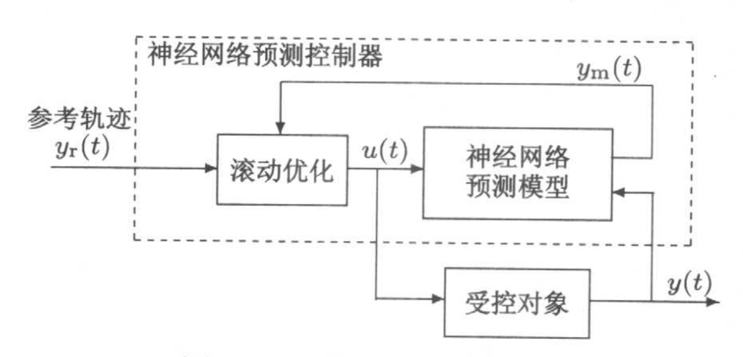


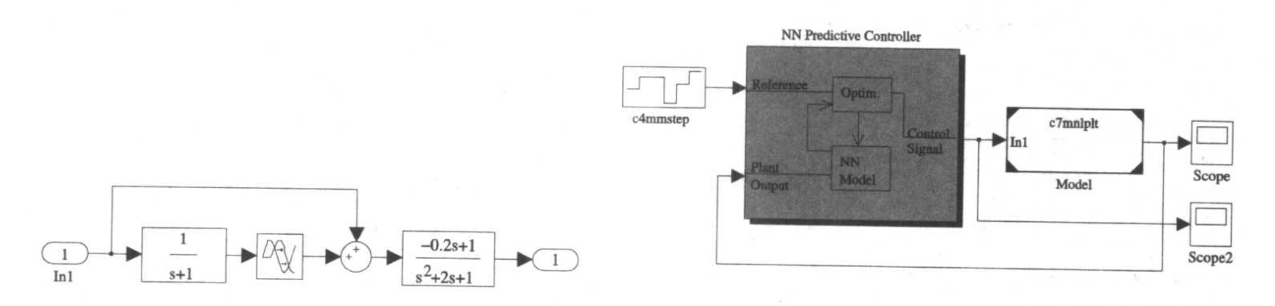
图 7-39 神经网络预测控制

MATLAB 的神经网络工具箱提供了可以用于 Simulink 仿真的神经网络预测控制器模块，可以直接用于任意复杂系统的预测控制。下面将通过例子演示该模块的使用方法及仿真方法。

例 7-15 考虑例 5-32 中给出的受控对象模型 $G(s)=\dfrac{(-0.2s+1)\left(1+\dfrac{e^{-s}}{s+1}\right)}{(s+1)^2}$，并期望 $|u(t)|\leqslant 5$，试设计出神经网络预测控制器，并得出控制效果。

求解 如果想使用神经网络预测控制模块，则可以采用下面步骤设计与仿真系统：

① **仿真模型搭建** 需要首先搭建起受控对象模型，在模型中分别用 In1 和 Out1 模块表示对象的输入输出信号，如图 7-40 (a) 所示。由这样搭建的受控对象模型可以搭建出控制仿真模型，如图 7-40 (b) 所示，其中使用了例 4-43 中构造的多阶梯信号作为输入信号。



(a) 受控对象模型 (文件名：c7mnlplt.mdl)　　(b) 控制仿真模型 (文件名：c7mnnmpc.mdl)

图 7-40 受控对象及神经网络模型预测控制系统仿真模型

② **控制器参数输入** 双击控制器模块，则得出如图 7-41 所示的控制器参数对话框。可以按照图中给出的数值填入各个时域及加权参数。当然，这些参数可以在以后的仿真中根据实际情况加以修正，以达到更好的控制效果。

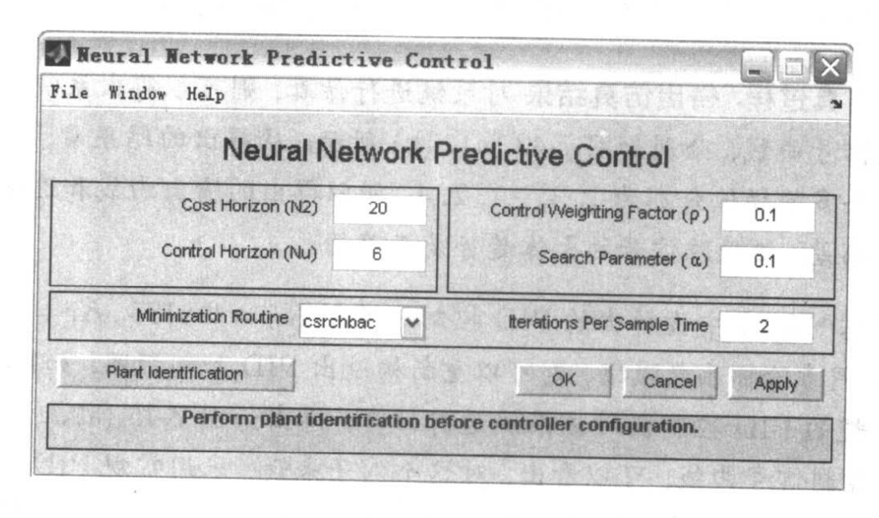
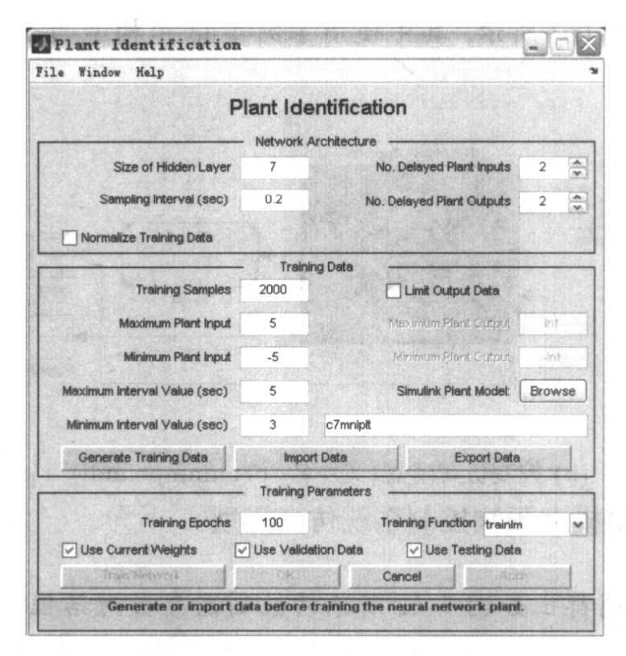


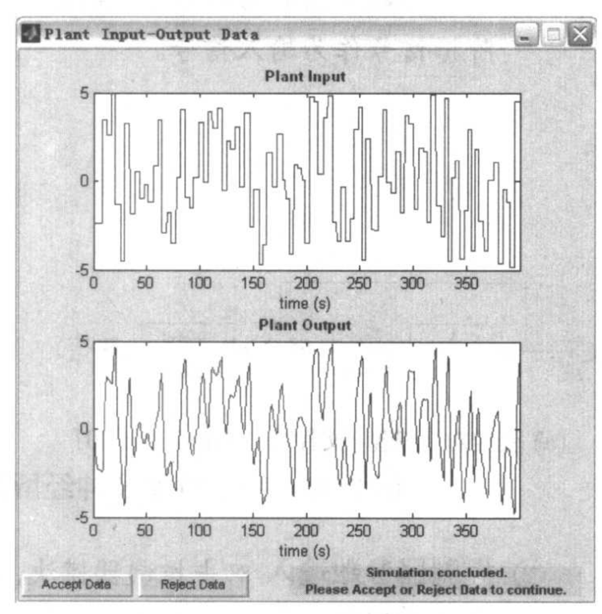
图 7-41 神经网络预测控制器参数对话框

③ **受控对象的神经网络近似** 应该建立起受控对象模型的神经网络近似。单击上述对话框中的 Plant Identification 按钮，则可以得出如图 7-42 (a) 所示的

对话框，在受控对象栏目 plant 下填入受控对象模型名 c7mnlplt，这样单击 Generate Training Data 按钮则可以由受控对象生成辨识用输入、输出信号，如图 7-42 (b) 所示。可以单击 Accept Data 接受获得的数据，也可以选择 Reject Data 按钮放弃数据，重新生成一组新数据。确认了辨识数据后，可以单击 Train Network 按钮来启动神经网络的训练过程，得出可以很好逼近受控对象模型的神经网络模型。



(a) 辨识参数对话框



(b) 生成的输入输出数据

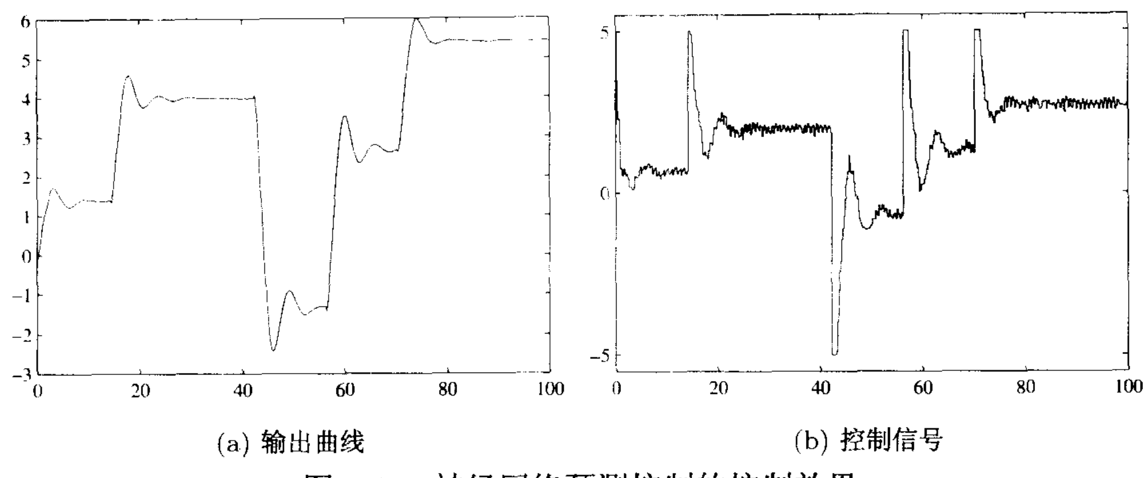
图 7-42 受控对象模型的神经网络近似

④ **启动仿真过程，得出仿真结果** 对系统进行仿真，则可以得出系统的输出曲线和控制信号曲线，分别如图 7-43 (a)、(b) 所示。从得出的结果看，由于期望输出的几个多阶梯值分别为 1, 3, -1, 2, 4，所以得出的输出曲线和这些设定值有着较大偏差。但跟踪信号的大体趋势是正确的。

重新考虑例 5-32 中设计出的 PID 控制器模型 $K_{\rm p}=0.5175$, $K_{\rm i}=0.1910$, $K_{\rm d}=0.1028$，若采用多阶梯函数激励，则可以重新构造出 PID 控制框图，如图 7-44 所示。

利用常规的 PID 控制器对该系统进行仿真，得出如图 7-45 (a)、(b) 所示的输出跟踪曲线和控制信号曲线，可以看出，对这个例子来说，采用常规 PID 控制器的控制效果远远优于前面介绍的神经网络预测控制系统。故在实际控制中，不应该过分追求“新”的控制器形式，而应该采用合适的控制器形式。传统的 PID 控制器在参数合理设置后，对于时不变受控对象的控制效果往往是令人满意的。

神经网络工具箱还提供了一个基于神经网络的模型参考控制器模块，该模块



(a) 输出曲线　　(b) 控制信号

图 7-43　神经网络预测控制的控制效果

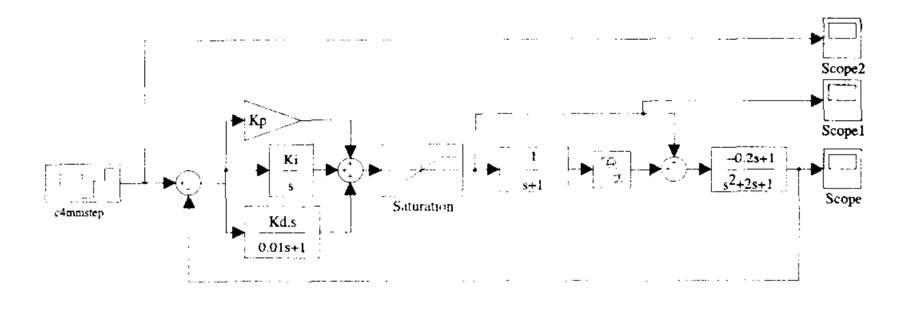
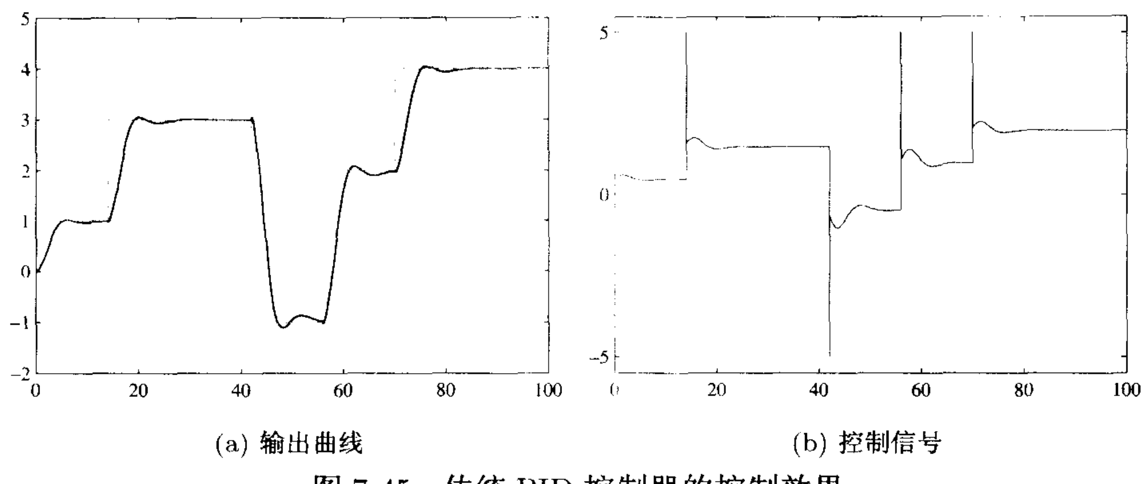


图 7-44　PID 控制系统仿真框图 (文件名：c7mopta.mdl)



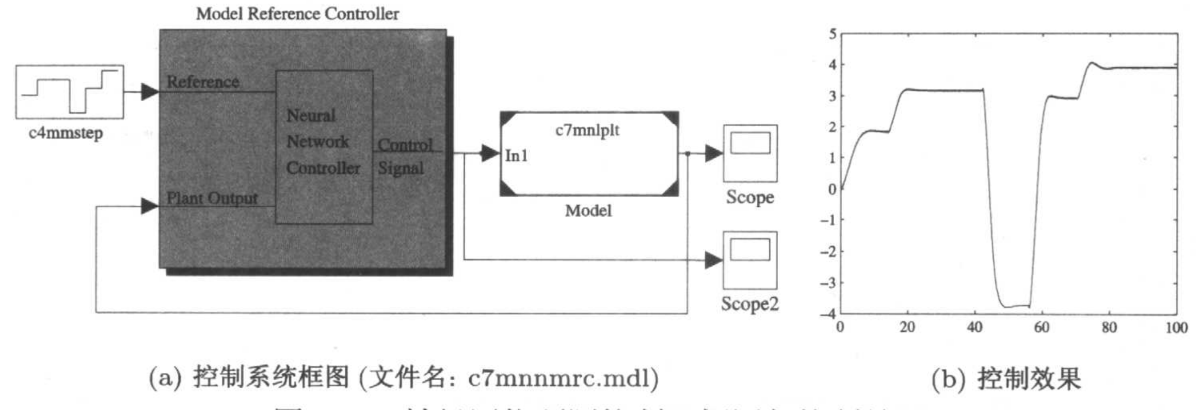
(a) 输出曲线　　(b) 控制信号

图 7-45　传统 PID 控制器的控制效果

首先用一个神经网络逼近受控对象模型，然后用另一个神经网络去实现控制器。下面仍然通过例子演示该控制器模块的使用方法和控制效果。

例 7-16 考虑例 7-15 中的控制问题，试用神经网络模型参考控制器研究其控制效果。

求解 用神经网络控制模块集中的神经网络模型参考模块代替例 7-15 框图中的控制器，如图 7-46 (a) 所示。可以对两个神经网络分别训练，具体方法类似于例 7-15，选定 Plant Identification 按钮训练受控对象神经网络，用该模块直接得出的界面训练控



(a) 控制系统框图 (文件名: c7mnnmrc.mdl) (b) 控制效果

图 7-46 神经网络预测控制器框图与控制效果

制器模型。训练控制器神经网络的方法也类似。训练完成后，就可以对原系统进行控制，得出如图 7-46 (b) 所示的控制效果。

从这里给出的例子可见，受控对象输出曲线和设定点之间存在较大的静态误差，在某些应用中不是很适合采用这类控制器。

7.3 基于进化方法的最优化计算与应用

第 5 章较详尽地介绍了各种各样的最优化问题求解方法。传统的最优化方法均从某个选定的初始点开始搜索最优解，所以难免出现局部最优值的情况。这里主要介绍两种基于进化的最优化方法 —— 遗传算法和粒子群算法，并给出基于 MATLAB 的最优化计算程序。从某种意义上讲，这样的算法更利于得出全局最优解。最后给出这样的优化算法在最优控制器设计中的应用。

7.3.1 遗传算法简介

遗传算法是基于进化论，在计算机上模拟生命进化机制而发展起来的一门新学科，它根据适者生存、优胜劣汰等自然进化规则来搜索和计算问题的解 [10,11]。该问题最早是由美国 Michigan 大学的 John Holland 于 1975 年提出的。遗传算法的基本思想是，从一个代表最优化问题解的一组初值开始进行搜索，这组解称为一个种群，种群由一定数量、通过基因编码的个体组成，其中每一个个体称为染色体，不同个体通过染色体的复制、交叉或变异又生成新的个体，依照适者生存的规则，个体也在一代一代进化，通过若干代的进化最终得出条件最优的个体。

MATLAB 7.0 版本开始提供新的遗传算法与直接搜索工具箱，可以较好地解决与遗传算法相关的各类问题。但早期版本没有官方的工具箱，只有两个基于 MATLAB 语言的免费遗传算法工具箱。一个是英国 Sheffield 大学自动控制与系统工程系 Peter Fleming 教授与 Andrew Chipperfield 开发的遗传算法工具箱，实现了各种基本运算，算法实现规范，说明书齐全，调用格式更类似于最优化工

具箱中的函数；另一个是美国北 Carolina 州立大学 Christopher Houck, Jeffery Joines 和 Michael Kay 开发的遗传算法最优化工具箱，其函数 gaopt() 可以直接解决最优化问题❶。对最优化问题来说，由于 GAOT 工具箱流传较广，已经能比较容易地解决最优化问题，所以这里还是建议采用该免费工具箱来解决基于遗传算法的最优化问题。

简单遗传算法的一般步骤为：

① 选择 N 个个体构成初始种群 $\boldsymbol{P}_0$，并求出种群内各个个体的函数值。染色体可以用二进制数组表示，也可以用实数数组来表示，种群可以由随机数生成函数建立。其实使用遗传算法求解函数 gaopt()，则会自动生成所需的初始种群 $\boldsymbol{P}_0$。

② 设置代数为 $i=1$，即设置其为第 1 代。

③ 计算选择函数的值，所谓选择即通过概率的形式从种群中选择若干个体的方式。遗传算法最优化工具箱提供了 3 个选择函数，其中 roulette() 实现了轮盘选择算法，normGeomSelect() 函数实现了归一化几何选择方法，tournSelect() 实现了锦标赛形式的选择方式，normGeomSelect() 函数为默认选择函数。

④ 通过染色体个体基因的复制、交叉、变异等创造新的个体，构成新的种群 $\boldsymbol{P}_{i+1}$，其中复制、交叉和变异都有相应的 MATLAB 函数，gaopt() 函数选择其中默认的方法进行这样的处理，构成新的种群。

⑤ $i=i+1$，若终止条件不满足，则转移到步骤③继续进化处理。

和传统最优化算法比较，遗传算法主要有以下几点不同[12]：

① 不同于从一个点开始搜索最优解的传统的最优化算法。遗传算法从一个种群开始对问题的最优解进行并行搜索，所以更利于全局最优化解的搜索，但遗传算法需要指定各个自变量的范围，而不像最优化工具箱中可以使用无穷区间的概念。

② 遗传算法并不依赖于导数信息或其他辅助信息来进行最优解搜索，而只由目标函数和对应于目标函数的适应度水平来确定搜索的方向。

③ 遗传算法采用的是概率性规则而不是确定性规则，所以每次得出的结果不一定完全相同，有时甚至会有较大的差异。

---

❶ 该工具箱的主函数名为 ga()，但该函数名与遗传算法和直接搜索工具箱中的函数同名，故这里将其改名为 gaopt()，原工具箱中其他函数也应该适当修改。

7.3.2 基于遗传算法的最优化问题求解

1. 求解函数的直接调用

这里将主要介绍遗传算法最优化工具箱中的 gaopt() 函数在求解最优化问题中的应用，介绍使用该函数的原因是因为该函数调用简单。即使对遗传算法理解不多，甚至不知道染色体如何选择，如何进行交叉和变异，如何进行选择等关于遗传算法的最基本知识，但利用 MATLAB 语言描述出目标函数，就可以得出最优解。和最优化工具箱不同，gaopt() 函数能求解的问题是最大化问题，所以在编写目标函数时应该注意。gaopt() 函数的常用调用格式为

```
[a,b,c]=gaopt(bound,fun)   % 最简调用格式
[x,b,c,d]=gaopt(bound,fun,p,v,P0,fun1,n)
```

其中，bound=[$x_{\rm m}$,$x_{\rm M}$] 为求解区间下界 $x_{\rm m}$ 和上界 $x_{\rm M}$ 构成的矩阵，fun 为字符串，表示用户编写的目标函数，其写法与最优化工具箱的目标函数写法相近，但结构不完全相同，后面将通过例子来描述之。返回的 $\boldsymbol{a}$ 为搜索的结果向量，由搜索出的最优 $\boldsymbol{x}$ 向量与目标函数构成，b 为搜索的最终种群，c 为搜索中间过程参数表，其第一列为代数，后面各列分别为该代最好的个体与目标函数的值，可以认为是寻优的中间结果。在第 2 种调用格式中，$\boldsymbol{p}$ 可以给目标函数增加附加参数，这些参数必须和目标函数对应，$\boldsymbol{v}$ 为精度及显示控制向量，$\boldsymbol{P}_0$ 为初始种群，fun1 为终止函数的名称，默认值为 'maxGenTerm'，n 为最大的允许代数。当然还有其他的调用参数的用户设置格式，如选择函数及参数、变异函数及参数等，在这里不具体介绍，读者可以参见文献 [13]。

MATLAB 7.0 版开始提供遗传算法与直接搜索工具箱，其中的 ga() 函数可以直接求解基于遗传算法的无约束最优化问题，而 MATLAB 7.1 版本推出了全新的遗传算法与直接搜索工具箱 2.0 版[14]，可以直接求解带有各种约束条件的最优化问题。和最优化工具箱中的相应函数类似，ga() 函数求解的仍然是最小化问题，其调用格式为

```
[x,f,flag,out]=ga(fun,n,opts)
[x,f,flag,out]=ga(fun,n,A,B,Aeq,Beq,xm,xM,nfun,opts)
```

其中，fun 为描述目标函数的 MATLAB 函数，其格式与最优化工具箱一致，但优化变量个数 n 为必须提供的变量，opts 为遗传算法控制选项，可以调用 gaoptimset() 函数设置各种选项。例如，用其 Generations 属性可以设定最大允许的代数，InitialPopulation 属性可以设置初始种群，用 PopulationSize 属性可以给定种群的规模，用 SelectionFcn 属性可以定义选择函数等等。函数调用结束后，返回的 $\boldsymbol{x}$ 为搜索的结果。若返回的 flag 大于 0，则表示求解成功，否则求解出现问题。

调用 MATLAB 7.1 中的 ga() 函数求解有约束最优化问题时，应采用 gaoptimset() 函数修改变异函数属性，并可以考虑增大初始种群的大小，如设置成 100

```
opts=gaoptimset('MutationFcn',@mutationadaptfeasible,...
    'PopulationSize',100); % 变异函数与种群函数设置
```

例 7-17 考虑一个简单的一元函数最优化问题求解，$f(x) = x\sin(10\pi x) + 2$，$x \in (-1, 2)$，试求出 $f(x)$ 取最大值时 x 的值。

求解 用下面的语句可以绘制出求解区间内的目标函数的曲线，如图 7-47 所示。可以看出，该曲线为振荡曲线，存在很多极值点。

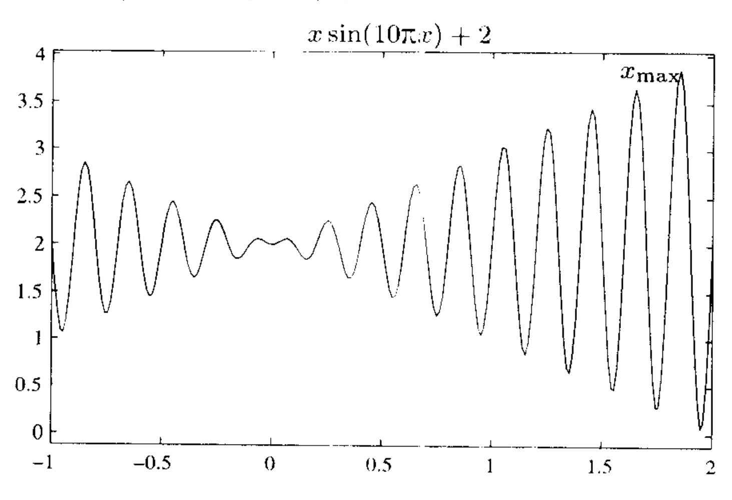


图 7-47 目标函数的曲线表示

```
>> ezplot('x*sin(10*pi*x)+2',[-1,2])
```

因为最优化工具箱的搜索函数需要给出初值，所以对不同的初值可能得出不同的搜索结果，例如可以给出如下的语句测试不同初值，得出的结果如表 7-5 所示。

```
>> f=@(x)[-x.*sin(10*pi*x)-2]; v=[];
   for x0=[-1:0.8:1.5,1.5:0.1:2]
      x1=fmincon(f,x0,[],[],[],[],-1,2); v=[v; x0,x1,f(x1)];
   end
```

可见，随意选择一个初值很难得出全局最优解，故用传统的寻优方式不一定能得出满意的结果。

利用遗传算法函数 gaopt()，完全选择默认选项，则可以编写一个文件描述目标函数如下：

```
function [sol,y]=c7mga1(sol,options)
x=sol(1); y=x.*sin(10*pi*x)+2;
```

这样，完全用默认参数调用遗传算法工具箱中的 gaopt() 函数，而不对其编码等

表 7-5 不同初值 x_0 下搜索到的最优解及目标函数值

x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$
−1	−1	−2	1.4	1.45070	−3.45035	1.7	1.25081	−3.25040
−0.2	−0.65155	−2.65078	1.5	0.25397	−2.25200	1.8	1.85055	−3.85027
0.6	0.65155	−2.65078	1.6	1.65061	−3.65031	1.9	0.452233	−2.451121

做任何指定，则可以得出如下的结果

```
>> [x0,b,fopt,d]=gaopt([-1,2],'c7mga1'); x0, fopt
```

可以得出 $x = 1.85054746647533, f_{\rm opt}(x_0) = 3.85027376676810$。可见，通过 100 代的搜索，可以得出相当高精度的寻优结果。

例 7-18 试求函数 $f(x) = (x_1 + x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$ 的最小值。

求解 可以按照遗传算法工具箱编写如下函数，描述最优化问题的目标函数。注意，这里应该描述成目标函数的最大值。

```
function [sol,f]=c7mga3(sol,options)
x=sol(1:4);
f=-(x(1)+x(2))^2-5*(x(3)-x(4))^2-(x(2)-2*x(3))^4-10*(x(1)-x(4))^4;
```

使用遗传算法函数 gaopt()，并设定自变量的求解范围为 $-1 \leqslant x_i \leqslant 1$，$i = 1,2,3,4$，则可以由下面的函数求解最优化问题，得出如下的结果，且部分中间结果如表 7-6 所示。事实上，该函数的精确解为 $x_1 = x_2 = x_3 = x_4 = 0$，但用遗传算法不能搜索到该点，只能得出次最优值。

```
>> [a,b,c,d]=gaopt([-1,1; -1 1; -1 1; -1 1],'c7mga3'); a,c
```

进化 100 代得出的解为 $\boldsymbol{x}^{\rm T}$ =[−0.00554, 0.00582, 0.0168, 0.0169]。

上述的求解结果显然误差很大，其最主要的原因是最大允许的代数 (默认值为 100) 太小。另外，求解的区间太小，使得得出解的可信度降低。也就是说，允许的求解区域是 $[-1, 1]$，如果选择更大的求解区域将得出更不精确的结果。

2. 修改函数调用的控制选项

现在考虑 gaopt() 函数稍复杂一点的调用格式，如下：

$[\boldsymbol{x}$,b,c,d]=gaopt(bound,fun,$\boldsymbol{p}$,$\boldsymbol{v}$,$\boldsymbol{P}_0$,fun1,n)

其中，$\boldsymbol{p}$ 可以给目标函数增加附加参数，这些参数必须和目标函数对应，$\boldsymbol{v}$ 为精度及显示控制向量，$\boldsymbol{P}_0$ 为初始种群，fun1 为终止函数的名称，默认值为 'maxGenTerm'，n 为最大的允许代数。当然还有其他的调用参数的用户设置格式，如选择函数及参数、变异函数及参数等，在这里不具体介绍，读者可以参见文

表 7-6 遗传算法搜索的部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
1	0.27677275	−0.033311652	−0.1791366	−0.15866754	−0.43203243
5	−0.02568998	−0.033311652	−0.1791366	−0.15866754	−0.01985436
9	−0.02568998	−0.033311652	−0.14097391	−0.12317016	−0.0097907197
13	−0.026182617	−0.002439155	0.038789725	0.018215623	−0.0030155289
19	−0.026182617	−0.002439155	0.018694947	0.018215623	−0.00086172757
25	−0.025312743	−0.002439155	0.018694947	0.018215623	−0.0008097327
30	−0.0079623182	0.0077997061	0.018694947	0.018215623	-6.6379887×10^{-6}
56	−0.0064495637	0.0063253498	0.018694947	0.018478891	-5.0417912×10^{-6}
86	−0.0068643431	0.0064421502	0.017019982	0.017036287	-4.0228234×10^{-6}
100	−0.0055401159	0.0058190087	0.01684682	0.016862616	-3.2016093×10^{-6}

献 [13]。

例 7-19 仍考虑例 7-18 中给出的最优化问题，试设置更多的允许代数，观察寻优的结果，并比较和最优化搜索算法得出结果在时间、精度上的差异。

求解 考虑新的求解区域，$-1000 \leqslant x_i \leqslant 1000$，由于求解范围增大，采用默认的代数 100 会有问题，所以可以考虑用新介绍的求解格式。显然，目标函数由 $\boldsymbol{x}$ 就能惟一确定，而无需附加参数，故令 p=[] 即可。同样，因为不想改变初始种群、控制向量等，可以将它们设置成空向量。这样，若想指定 2000 代为最大进化代数，则可以给出如下的命令，得出的中间结果如表 7-7 所示。

```
>> tic, xmM=[-ones(4,1),ones(4,1)]*1000;
   [a,b,c,d]=gaopt(xmM,'c7mga3',[],[],[],'maxGenTerm',2000);
   a(1:4), dd=[c(1:100:end,:); c(end,:)], toc
```

耗时 13.94 秒，得出最优解 $\boldsymbol{x}^{\mathrm{T}}=[0.00839, -0.00839, -0.0097, -0.0097]$。

可见，在 1425 代左右的误差都一直很大，所以用默认的 100 代处理不了这样大求解区间的问题，必须增大代数。然而若已达到较高求解精度，则再进一步增加最大允许的代数也不会显著改善求解的精度了。

现在考虑用第 5 章的无约束最优化求解功能，由如下的命令直接求出原问题的解，其精度明显高于遗传算法的结果，且求解时间大大减少。

```
>> f=@(x)(x(1)+x(2))^2+5*(x(3)-x(4))^2+...
        (x(2)-2*x(3))^4+10*(x(1)-x(4))^4;
   ff=optimset; ff.MaxIter=10000; ff.TolX=1e-7;
```

表 7-7　遗传算法搜索的部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
1	−900.54261	832.93703	364.5536	−814.31522	-6.7599172×10^8
211	−878.91097	−578.75001	−293.87781	−873.72917	−3819700.7
471	−787.67855	−590.90348	−299.86119	−786.66609	−3091442.6
664	−262.06476	−512.5168	−258.38886	−264.75434	−601032.09
851	−124.98415	−424.5766	−213.45722	−127.61195	−339370.78
1054	−35.211178	−171.93765	−86.407361	−37.414245	−55148.419
1219	−2.8405531	−3.4714646	−2.4158781	−3.1699584	−46.226448
1425	−0.42995364	−0.79729127	−0.75195129	−0.74178084	−1.8504961
1690	−0.0099107861	−0.037154103	−0.1631374	−0.15852548	−0.014186929
1906	0.0038080427	−0.0075509811	−0.064703926	−0.064171806	−0.00044948145
2000	0.0083858532	−0.0083918002	−0.0096953507	−0.0096903034	-1.0824368×10^{-6}

```
tic, x=fminsearch(f,10*ones(4,1),ff); toc; x'
```

该求解语句耗时 0.18 秒，得出的解为

$$\boldsymbol{x}^{\rm T}=[3.04\times10^{-8},-3.04\times10^{-8},-7.53\times10^{-7},-7.53\times10^{-7}]$$

3. 有约束最优化问题的求解方法

从传统意义上讲，基于进化类算法求解最优化问题都是针对无约束最优化问题而言的，在实际应用中又经常遇到有约束的最优化问题，所以将该方法扩展到有约束问题的求解是很有必要的。考虑如下的有约束最优化问题

$$J=\max_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}\boldsymbol{g}(\boldsymbol{x})\leqslant\boldsymbol{0}\\ \boldsymbol{h}(\boldsymbol{x})=\boldsymbol{0}\\ \boldsymbol{x}_{\rm m}\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_{\rm M}\end{array}\right.} f(\boldsymbol{x}) \tag{7-3-1}$$

对于不等式约束条件来说，可以在描述目标函数时将不满足约束条件的 $\boldsymbol{x}$ 向量处的函数值人为地设置成很小的值，迫使最优化搜索排除该点。等式约束条件的处理稍微麻烦一些，因为等式约束条件的存在，通过求解 m 个非线性方程 $\boldsymbol{h}(\boldsymbol{x})=\boldsymbol{0}$ 将原来的 n 元函数最优化问题等效地变换成 $n-m$ 元的问题。这样，原来不容易求解的有约束最优化问题就可以变换成无约束最优化问题来直接求解了。

例 7-20 试用遗传算法求解下面的线性规划问题

$$\min_{x \text{ s.t.} \begin{cases} -2x_1+x_2+x_3\leqslant 9 \\ -x_1+x_2\geqslant -4 \\ 4x_1-2x_2-3x_3=-6 \\ x_{1,2}\leqslant 0, x_3\geqslant 0 \end{cases}} (x_1+2x_2+3x_3)。$$

求解 由等式约束可以解出 $x_3=(6+4x_1-2x_2)/3$。可见，原来三元最优化问题可以转换成二元最优化问题。由上面的推导，可以用下面的 MATLAB 函数描述目标函数为

```
function [sol,y]=c7mga4(sol,options)
x=sol(1:2); x=x(:); x(3)=(6+4*x(1)-2*x(2))/3;
y1=[-2 1 1]*x; y2=[-1 1 0]*x;
if (y1>9 | y2<-4 | x(3)<0), y=-100; else, y=-[1 2 3]*x; end
```

其中用 x_1, x_2 数据计算出 x_3 的值，并判定约束条件是否满足，在不满足约束条件时人为地将目标函数的值设置成 -100，有意排除这些个体，就能由下面语句较好地解决有约束最优化问题。

```
>> [a,b,c]=gaopt([-1000 0; -1000 0],'c7mga4',[],[],[],...
        'maxGenTerm',1000);
   c=[c(1:15:end,:); c(end,:)]; a,c
```

这里只得出了 x_1, x_2，而 $x_3=(6+4x_1-2x_2)/3=0.00005529863957$。遗传算法的中间结果由表 7-8 给出。

表 7-8 遗传算法搜索的部分中间结果

代	x_1	x_2	$f(\boldsymbol{x})$	代	x_1	x_2	$f(\boldsymbol{x})$
1	−49.87071	−205.1789	−100	561	−6.671794	−10.42314	27.35897
60	−1.542377	−2.79424	1.711883	636	−6.951279	−10.92534	28.75639
121	−3.301757	−4.363803	10.50878	671	−6.958811	−10.92531	28.79406
155	−4.776542	−6.982296	17.88271	697	−6.958938	−10.92996	28.79469
221	−5.011393	−7.813805	19.05696	750	−6.965209	−10.93186	28.82605
300	−5.661174	−9.35381	22.30587	876	−6.966025	−10.9325	28.83012
385	−5.995186	−9.018754	23.97593	945	−6.997332	−10.99666	28.98666
421	−6.383564	−9.945158	25.91782	980	−6.998994	−10.99808	28.99497
458	−6.407366	−9.981396	26.03683	1000	−6.999769	−10.99962	28.99885

其实，该问题用线性规划函数可以立即得出更精确的结果。

```
>> f=[1 2 3]; A=[-2 1 1; 1 -1 0]; B=[9; 4]; Aeq=[4 -2 -3]; Beq=-6;
   x=linprog(f,A,B,Aeq,Beq,[-inf;-inf;0],[0;0;inf]); x'
```

将得出 $x_1=-6.99999999999, x_2=-10.99999999999$，可见，由传统方法得出的结果

更精确。

从最优化问题求解的方法看，最优化工具箱中的函数一次只能搜索到一个解，对非凸性问题来说往往可能找到一个局部最优值，而用遗传算法则可以同时从一组初值点出发，有可能找到更好的局部最优值甚至全局最优值，但其求取最优值算法的精度和速度均不是很理想。在实际求解问题中，可以考虑采用这样的策略，先用遗传算法初步定出较好最优值所在的大概位置，然后以该位置为初值，调用最优化工具箱中的函数快速、准确地求出该最优值。

7.3.3　粒子群优化算法及 MATLAB 求解

粒子群优化 (particle swarm optimization，PSO) 算法是文献 [15] 提出的一种进化算法，该算法是受生物界鸟群觅食的启发而提出的搜索食物，即最优解的一种方法。假设某个区域内有一个食物 (全局最优点)，有位于随机初始位置的若干个鸟（或粒子），每一个粒子有到目前为止自己的个体最优值 $p_{i,\mathrm{b}}$，整个粒子群有到目前为止群体的最优值 g_b，这样每个粒子可以根据下面的式子更新自己的速度和位置

$$v_i(k+1)=\phi(k)v_i(k)+\alpha_1\gamma_{1i}(k)[p_{i,\mathrm{b}}-x_i(k)]+\alpha_2\gamma_{2i}(k)[g_\mathrm{b}-x_i(k)] \tag{7-3-2}$$

$$x_i(k+1)=x_i(k)+v_i(k+1) \tag{7-3-3}$$

其中 γ_{1i} 和 γ_{2i} 为 $[0,1]$ 区间内均匀分布的随机数，$\phi(k)$ 为惯量函数，α_1 和 α_2 为加速常数。

文献 [16] 给出了一个基于粒子群优化算法的 MATLAB 工具箱[❶]，其主函数 `pso_Trelea_vectorized()` 可以用来搜索最优值，该函数的常用调用格式为

$$\texttt{[sol,tr]=pso_Trelea_vectorized(fun,}n\texttt{,}v_\mathrm{M}\texttt{,}[\boldsymbol{x}_\mathrm{m},\boldsymbol{x}_\mathrm{M}]\texttt{,key,options)}$$

其中 `fun` 为描述目标函数的 MATLAB 函数名，n 是 $\boldsymbol{x}$ 向量的维数，这两个量是求解最优化问题必须提供的，其他的变量是可选的；变量 v_M 是最大允许的速度，其默认值为 4；$\boldsymbol{x}_\mathrm{m},\boldsymbol{x}_\mathrm{M}$ 是每个变量的最小和最大值向量，默认为 ± 100；`key` 为极值类型选择，默认的 0 为最小值，取 1 表示求最大值；选项 `options` 为结构体控制变量；返回的 $(n+1)\times 1$ 列向量 `sol` 由两个部分构成，前 n 个元素为搜索到的 $\boldsymbol{x}$ 向量，第 $n+1$ 个元素是目标函数的最优值；返回的 `tr` 记录了寻优的中间结果。主函数基于 Trelea 算法[17]实现了粒子群优化功能。该函数调用了神经网络工具箱，并将寻优的中间训练过程用图形显示出来，比较直观。该函数还支持 Clerc 算法[18]。

---

❶ 工具箱地址：`http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=7506&objectType=file`。

这里给出的函数是“向量化”版本，允许一次性运行一组粒子向量的目标函数求值，故其效率远远高于非向量化的版本。在目标函数的描述上与传统的最优化目标函数有不同之处。前面介绍的目标函数中 $\boldsymbol{x}$ 为向量，每个元素对应 x_i 的当前值，而这里的 $\boldsymbol{x}$ 为矩阵，其第 i 列为各个粒子的 x_i 值，所以在原来目标函数程序中用的 $x(i)$ 变量应该修改为 $x(:,i)$，且相应地应该进行点运算。下面将用实际例子演示最优化求解方法。

例 7-21 试用粒子群算法求解例 7-18 中给出的无约束最优化问题。

求解 编写一个 MATLAB 函数描述目标函数

```
function f=c7mpso1(x)
f=(x(:,1)+x(:,2)).^2 + 5*(x(:,3)-x(:,4)).^2 +...
    (x(:,2)-2*x(:,3)).^4 + 10*(x(:,1)-x(:,4)).^4;
```

注意，在目标函数中的向量化表示和点运算，如果去掉这些表示，程序将无法运行。可以按照遗传算法工具箱编写如下函数，描述最优化问题的目标函数。注意，这里应该描述

```
>> x=pso_Trelea_vectorized('c7mpso1',4); x(1:4)
```

由前面的命令得出的最优解近似值为 $\boldsymbol{x}=[-0.00059, 0.00059, 0.00227, 0.00227]^{\mathrm{T}}$，解的图示由图 7-48 给出。

例 7-22 重新考虑例 7-20 中给出的有约束最优化问题，试用粒子群算法求解该问题。

求解 类似于前面基于遗传算法程序的编写方法，注意这里采用的是向量化的模式描述目标函数，可以编写出如下的函数

```
function y=c7mpso4(x)
x1=x(:,1); x2=x(:,2); x3=(6+4*x1-2*x2)/3; x=[x x3]';
y1=[-2 1 1]*x; y2=[-1 1 0]*x; y=[1 2 3]*x;
ii=find(y1>9|y2<-4|x3'<0); y(ii)=100; y=y(:);
```

由下面的语句求解该问题，得出精确的最优解 $\boldsymbol{x}^{\mathrm{T}}=[-7,-11,0]$。该函数调用后将得出类似于图 7-48 所示的搜索结果。

```
>> x=pso_Trelea_vectorized('c7mpso4',2);
   [x(1:2); (6+4*x(1)-2*x(2))/3]
```

7.3.4 基于遗传算法的最优控制问题求解

从前面介绍的遗传算法应用看，其优势在于能求解最优化问题的全局最优解，所以可以考虑在系统设计中引入遗传算法作为解决问题的工具。下面将通过例子演示遗传算法在最优控制器设计中的应用，并指出该方法不能应用的场合。

例 7-23 考虑不稳定受控对象 $G(s)=\dfrac{s+2}{s^4+8s^3+4s^2-s+0.4}$，试为其设计一个最优

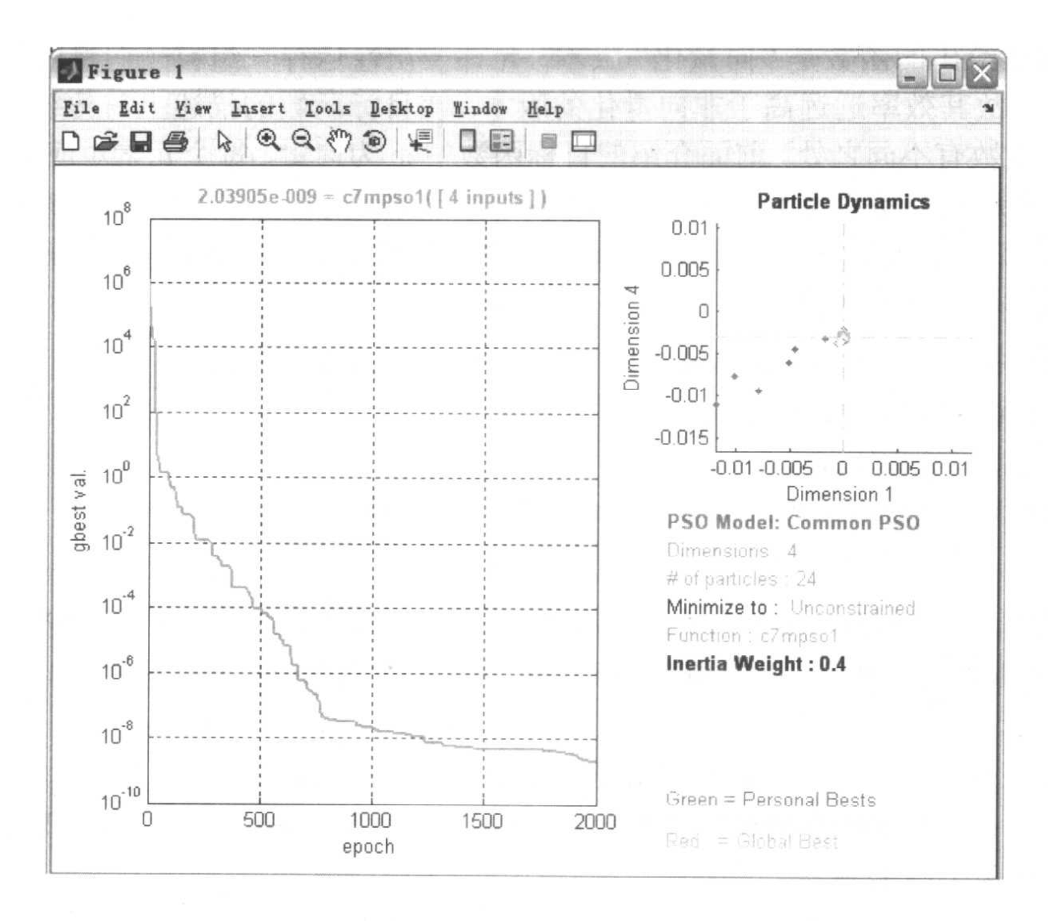


图 7-48 粒子群优化算法寻优结果的图示

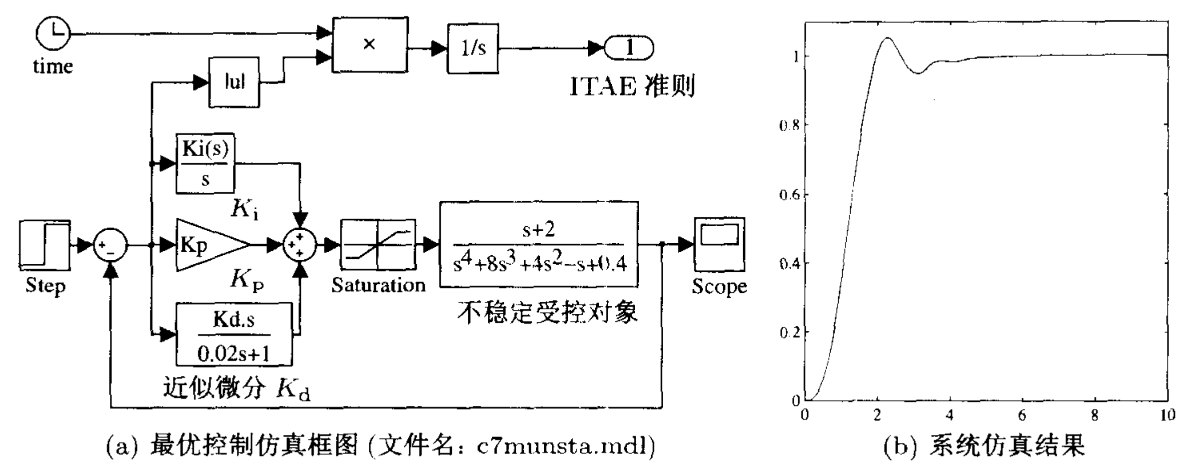
PID 控制器，使得 ITAE 准则的值最小。

求解 前面几节求解最优化问题的方法是，首先给出一个初值，然后通过搜索的方法从初值点开始搜索最优值。然而，对不稳定受控对象来说，这种算法的最大问题是不容易选择出能使得闭环系统稳定的控制器初值，故常规最优化算法不能正常启动。若想为其设计最优控制器，可以考虑采用遗传算法一次性多选择一些初始值，以此作为初始种群，开始最优解的求解。同时选择多个初值的好处是，在众多的初始个体中，往往可能有使得闭环系统稳定的个体，故遗传算法有望能得出最优控制器。

求解最优控制器设计问题，可以首先由 Simulink 搭建出如图 7-49 (a) 所示的仿真框图。其中，为了不使控制信号过大，在 PID 控制器后串一个饱和非线性环节，使得饱和区域为 $\Delta=5$，由该框图按照遗传算法优化工具箱的目标函数定义方法可以写出下面的 MATLAB 函数来描述。此最优化问题的目标函数

```
function [sol,y]=c7funun(x,options)
sol=x; assignin('base','Kp',x(1));
assignin('base','Kd',x(2)); assignin('base','Ki',x(3));
[t_time,x_state,y_out]=sim('c7munsta.mdl',[0,5]); y=-y_out(end,1);
```

注意，这里定义的目标函数是取极大值的，所以在 OCD 标准定义上加了负号。



(a) 最优控制仿真框图 (文件名：c7munsta.mdl)　　(b) 系统仿真结果

图 7-49　不稳定受控对象的最优 PID 控制器设计

现在假设想在 (0.1, 100) 范围内搜索该问题的最优解，则可以在 MATLAB 工作空间中输入下面的命令。

```
>> ctrl_pars=gaopt([0.1*ones(3,1), 100*ones(3,1)],'c7funun'),
   c7funun(ctrl_pars(1:3))      % 显示最终控制曲线
```

这样可以设计出最优控制器为 $G_c(s)=47.8313+\dfrac{0.2041}{s}+\dfrac{55.3632s}{0.01s+1}$，在该控制器的作用下的阶跃响应如图 7-49 (b) 所示，可见，对不稳定受控对象仍能利用遗传算法来设计出最优控制器，从而很好地控制受控对象。值得注意的是，控制器的积分分量参数很小，所以可以忽略，改用 PD 控制器也能得出较好的控制效果。

7.4　迭代学习控制的仿真研究

在机器人、硬盘驱动器伺服控制等诸多领域中，需要机器经常一次又一次地重复某个操作。在实际控制中人们很自然地会问这样的问题："在我们重复做某一个工作时，每次我们都学到一些东西，使得工作越做越好。机器控制在做同样工作时能不能也做到这点"？让控制器本身具有某种"智能"，使得它在控制过程中能不断完善自己，使得控制效果越来越好，这种具有"学习"能力的控制器就是本节需要探讨的内容。1978 年 Uchiyama 提出一个控制高速运动机械的思想，1984 年 Arimoto 发展了 Uchiyama 的思想，提出了迭代学习控制 (Iterative Learning Control，ILC) 的概念[19]。与鲁棒控制一样，ILC 也能处理实际动力学系统中的不确定性，但它能实现完全跟踪，控制器形式更为简单且需要较少的先验知识。这种控制方法适合于某种具有重复运动性质的被控对象，前几次运行可能有较大的控制误差，但通过"边干边学"的方式，逐步学习控制经验，利用系统前次的控制经验和输出误差来修正当前的控制作用，使系统输出尽可能收敛于期望值。ILC 的研究对具有较强的非线性耦合、较高的位置重复精度、难以建模和高精度轨迹

跟踪控制要求的动力学系统有着非常重要的意义。

文献 [20~23]对 ILC 控制算法和理论以及存在的问题和研究方向做了综述。本节介绍 ILC 基本原理、ILC 算法满足的条件。然后将介绍 ILC 算法的学习律及其在 MATLAB 语言中的设计方法。

7.4.1 迭代学习控制的基本原理

和前面介绍的很多内容相似，这里先写出受控对象的状态方程模型

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ \boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \end{cases} \tag{7-4-1}$$

其中 $\boldsymbol{x}$ 为 n 维状态向量；系统为 m 路输入，r 路输出；$\boldsymbol{f}(\cdot), \boldsymbol{g}(\cdot)$ 为相应维数的向量函数。迭代学习控制问题可以描述为：给定期望输出 $\boldsymbol{y}_\mathrm{d}(t)$，存在与之相应的期望输入 $\boldsymbol{u}_\mathrm{d}(t)$ 和每次运行的初始状态 $\boldsymbol{x}_k(0)$，要求通过多次重复运行，在给定的学习律下使系统在控制周期 $t \in [0, T]$ 内，获得系统控制输入序列 $\boldsymbol{u}_\mathrm{d}(t)$，使得系统输出 $\boldsymbol{y}_k(t)$ 能逼近期望的系统输出 $\boldsymbol{y}_\mathrm{d}(t)$。迭代学习控制的学习律一般可由递推的形式表示为

$$\boldsymbol{u}_k(t) = \mathscr{L}(\boldsymbol{u}_{k-1}(t), \boldsymbol{e}_{k-1}(t)) \tag{7-4-2}$$

其中 $\mathscr{L}(\cdot)$ 为线性或非线性算子，$\boldsymbol{e}_{k-1}(t) = \boldsymbol{y}_\mathrm{d}(t) - \boldsymbol{y}_{k-1}(t)$ 为前次运行的输出误差。由式 (7-4-2) 给出的学习律可见，开环迭代学习控制的基本原理是利用上一个工作周期内的误差信号 $\boldsymbol{e}_{k-1}(t)$ 和上一个周期的输入序列 $\boldsymbol{u}_{k-1}(t)$ 来构造本周期控制序列 $\boldsymbol{u}_k(t)$ 的方法。

当 k 足够大时，如果 $\boldsymbol{e}_{k-1}(t)$ 在 $t \in [0, T]$ 上一致趋于零，则称上述迭代学习控制是收敛的。收敛性问题是迭代学习控制中最重要的问题，只有迭代学习过程是收敛的，迭代学习控制才有实际应用意义。

开环迭代学习控制的基本原理如图 7-50 所示。在实际应用中，系统下一次运行的新的控制既可以在上一次运行结束后离线计算得到，也可以在上一次运行中在线计算得到；新的控制量存入存储器，刷新旧控制量。在施加控制时，需从存储器中取出控制量。如果取出的是当前的误差信号，则可以构造出闭环控制方式。可以看到，迭代学习控制算法可利用的信息要多于常规的反馈控制算法，它包括以前每次运行的所有时间段上的信息和当次运行的当前时刻之前时间段的信息。

在传统的迭代学习控制研究中，一般总是假定下述假设条件满足[24,25]：

① 系统每次运行时间间隔是有限的固定间隔，即 $[0, T]$；

② 在运行区间 $[0, T]$ 内系统的期望轨迹总是事先已知的；

③ 系统的初始条件重复，即每次运行前，系统的初始状态 $\boldsymbol{x}_k(0)$ 都相同；

④ 系统的动态结构在每次重复迭代运行中保持不变；

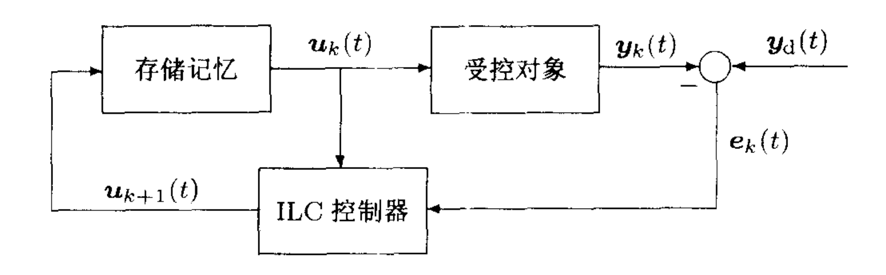


图 7-50 开环迭代学习控制基本结构

⑤ 系统每次运行的输出 $\boldsymbol{y}_k(t)$ 可测，且跟踪误差信号 $\boldsymbol{e}_k(t) = \boldsymbol{y}_\mathrm{d}(t) - \boldsymbol{y}_k(t)$ 可以用于构造下一个时刻的控制信号 $\boldsymbol{u}_{k+1}(t)$；

⑥ 系统的动态特性是可逆的，即对一个事先给定的输出轨迹 $\boldsymbol{y}_\mathrm{d}(t)$，存在惟一的控制信号 $\boldsymbol{u}_\mathrm{d}(t)$，使得系统产生理想的输出 $\boldsymbol{y}_\mathrm{d}(t)$。

一个成功的迭代学习控制算法不仅应该在每一次控制作用于系统后，能使得系统的输出误差变小，还需要有较快的收敛速度以保证算法的实用性。另外，迭代学习控制算法的收敛性应与具体的期望轨迹无关，如果给出一个新的期望轨迹，迭代学习律应该无需作任何改变即可使用。

7.4.2 迭代学习控制算法

1. 连续时间 PID 型 ILC 算法

如果受控对象为连续模型，则可以写出开环形式的和闭环形式的 PID 型迭代学习控制算法。

① **开环控制器** ILC 最初的概念是以开环的形式给出的，常见算法为 PID 型，开环 ILC 是指控制律中不包含当前过程信息的算法。连续时间 PID 型 ILC 算法的一般形式可以写成

$$\boldsymbol{u}_k(t) = \boldsymbol{u}_{k-1}(t) + \boldsymbol{\Gamma}_\mathrm{p}\boldsymbol{e}_{k-1}(t) + \boldsymbol{\Gamma}_\mathrm{i}\int_0^t \boldsymbol{e}_{k-1}(\tau)\mathrm{d}\tau + \boldsymbol{\Gamma}_\mathrm{d}\dot{\boldsymbol{e}}_{k-1}(t) \tag{7-4-3}$$

其中，$\boldsymbol{\Gamma}_\mathrm{p}, \boldsymbol{\Gamma}_\mathrm{i}, \boldsymbol{\Gamma}_\mathrm{d}$ 分别为比例、积分和微分相应维数的学习增益矩阵。如果其中某个或某些增益矩阵等于零，则可以简化为 P 型、D 型、PI 型、ID 型和 PD 型 ILC 控制器。

② **闭环控制器** 闭环 PID 型学习策略是取第 k 次运行的误差作为学习的修正项，从而有下面的学习律

$$\boldsymbol{u}_k(t) = \boldsymbol{u}_{k-1}(t) + \boldsymbol{\Gamma}_\mathrm{p}\boldsymbol{e}_k(t) + \boldsymbol{\Gamma}_\mathrm{i}\int_0^t \boldsymbol{e}_k(\tau)\mathrm{d}\tau + \boldsymbol{\Gamma}_\mathrm{d}\dot{\boldsymbol{e}}_k(t) \tag{7-4-4}$$

比较开环控制和闭环控制两种控制策略，可见开环控制是利用上一步的控制与误差信息计算控制规律，而闭环控制采用的是由上一步的控制和当前误差信息计算控制规律的方法。一般情况下，闭环迭代学习控制优于开环模式。

2. 离散时间 PID 型 ILC 算法

假设离散时间系统的受控对象模型为

$$\begin{cases} \boldsymbol{x}(t+1)=\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),t) \\ \boldsymbol{y}(t)=\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),t) \end{cases} \tag{7-4-5}$$

第 k 次运行时可表示为

$$\begin{cases} \boldsymbol{x}_k(t+1)=\boldsymbol{f}(\boldsymbol{x}_k(t),\boldsymbol{u}_k(t),t) \\ \boldsymbol{y}_k(t)=\boldsymbol{g}(\boldsymbol{x}_k(t),\boldsymbol{u}_k(t),t) \end{cases} \tag{7-4-6}$$

输出误差可以定义为 $\boldsymbol{e}_k(t)=\boldsymbol{y}_{\rm d}(t)-\boldsymbol{y}_k(t)$，根据该误差定义可以分别定义出开环和闭环 PID 型学习律为

$$\boldsymbol{u}_{k+1}(t)=\boldsymbol{u}_k(t)+\boldsymbol{\Gamma}_{\rm p}\boldsymbol{e}_k(t+1)+\boldsymbol{\Gamma}_{\rm i}\sum_{j=0}^{t+1}\boldsymbol{e}_k(j)+\boldsymbol{\Gamma}_{\rm d}\Big[\boldsymbol{e}_k(t+1)-\boldsymbol{e}_k(t)\Big] \tag{7-4-7}$$

$$\boldsymbol{u}_{k+1}(t)=\boldsymbol{u}_k(t)+\boldsymbol{\Gamma}_{\rm p}\boldsymbol{e}_{k+1}(t)+\boldsymbol{\Gamma}_{\rm i}\sum_{j=0}^{t}\boldsymbol{e}_{k+1}(j)+\boldsymbol{\Gamma}_{\rm d}\Big[\boldsymbol{e}_{k+1}(t)-\boldsymbol{e}_{k+1}(t-1)\Big] \tag{7-4-8}$$

当然，对连续受控对象模型也能采用离散的 ILC 算法。如果受控对象模型为线性单变量模型，则根据前面介绍的 PID 型开环 ILC 算法，可以编写出如下的 MATLAB 函数

```
function [y,e,u,kvec,t0]=ilc_lsim(G,T,kmax,yd,Kp,Ki,Kd)
n=length(yd); y=zeros(n,kmax); e=y; t0=0:T:(n-1)*T;
u=y; e(:,1)=yd(:); kvec=[];
if G.Ts==0, G=c2d(G,T); end; G=ss(G);
for k=1:kmax, x0=zeros(size(G.a,1),1);
    for t=2:n
       x1=G.a*x0+G.b*u(t,k);
       y(t,k)=G.c*x1+G.d*u(t,k);
       x0=x1; e(t,k)=yd(t)-y(t,k);
       u(t,k+1)=u(t,k)+Kp*e(t,k)+...   % P 控制
              Kd*(e(t,k)-e(t-1,k))+... % D 控制
              Ki*sum(e(1:t,k));        % I 控制
    end,
    kvec(k)=max(abs(e(:,k)));
end
```

该函数的调用格式为

$[\boldsymbol{y},\boldsymbol{e},\boldsymbol{u},\boldsymbol{k},\boldsymbol{t}]$=ilc_lsim$(G,T,k_{\rm max},\boldsymbol{y}_{\rm d},K_{\rm p},K_{\rm i},K_{\rm d})$

其中，G 为线性系统的数学模型，可以为连续的，也可以为离散的；T 为控制器的

采样周期；$k_{\max}$ 为最大迭代次数；$\boldsymbol{y}_\mathrm{d}$ 为参考轨迹向量；$K_\mathrm{p}, K_\mathrm{i}, K_\mathrm{d}$ 为 PID 控制参数，分别对应于算法中的 $\Gamma_\mathrm{p}, \Gamma_\mathrm{i}, \Gamma_\mathrm{d}$ 的值。返回的 $\boldsymbol{y}$ 为矩阵，每一列对应该次迭代的输出向量；$\boldsymbol{e}$ 和 $\boldsymbol{u}$ 亦为每次迭代中的误差和输入向量构成的矩阵；$\boldsymbol{k}$ 为每次跟踪控制误差信号的范数；$\boldsymbol{t}$ 为时间向量。

该函数适用于受控对象相对阶为 1 的情况，若相对阶大于 1，则应该考虑含有高阶微分项的 ILC 控制器[26]。

例 7-24 考虑离散对象的模型

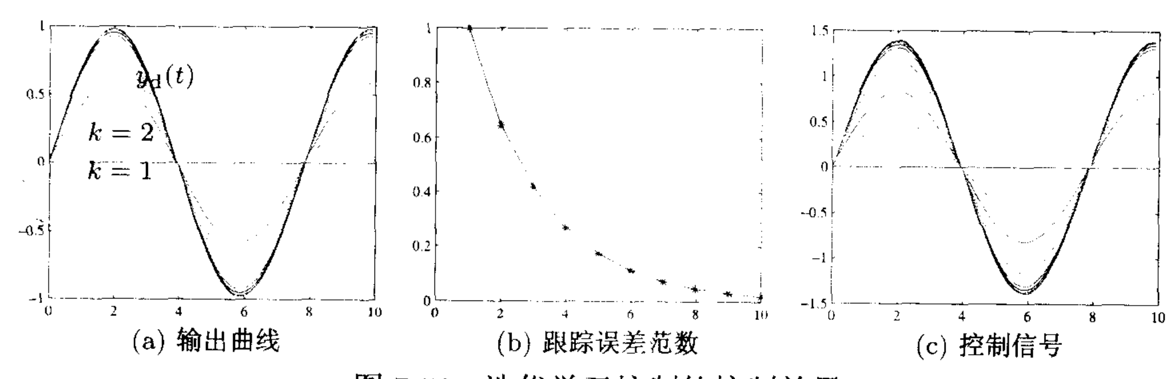
$$\boldsymbol{x}(k+1) = \begin{bmatrix} -0.8 & -0.22 \\ 1 & 0 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \boldsymbol{u}(k),\ \ y(k) = [1,\ 0.5]\boldsymbol{x}(k)$$

参考轨迹为 $y_\mathrm{d}(t) = \sin(0.08t)$，假设系统的初始状态为零，试设计出ILC 控制器。

求解 对 Arimoto P 型 ILC 控制器来说，若选择 $K_\mathrm{p} = 0.5$，$K_\mathrm{i} = K_\mathrm{d} = 0$，则由下面的语句可以仿真出迭代学习控制的控制效果

```
>> yd=sin(0.08*[0:99])';
   G=ss([-0.8,-0.22; 1 0],[0.5;1],[1 0.5],0,'Ts',0.1);
   [y,e,u,kvec,t]=ilc_lsim(G,0.1,10,yd,0.5,0,0);
   plot(t,y), figure, plot(kvec,'-*'), figure, plot(t,u)
```

得出的输出曲线如图 7-51 (a) 所示。由控制结果看，$k = 1$ 时系统没有开始响应，当 $k = 2$ 时，输出开始试图跟踪期望的信号，但有一定误差。控制器通过误差学习了如何控制该受控对象，所以当 $k = 3$ 时控制误差明显减小，但仍然存在误差，所以控制器进一步学习，得出 $k = 4$ 时的控制修改。迭代学习控制器正是通过这样的学习方式，取得了一次比一次好的控制效果，当 $k = 10$ 时基本接近于期望的轨迹。从图 7-51 (b) 给出的跟踪误差范数看，$k = 10$ 时仍有一定误差，所以该学习算法的 K_p 值不理想。每次控制计算出来的控制信号如图 7-51 (c) 所示。

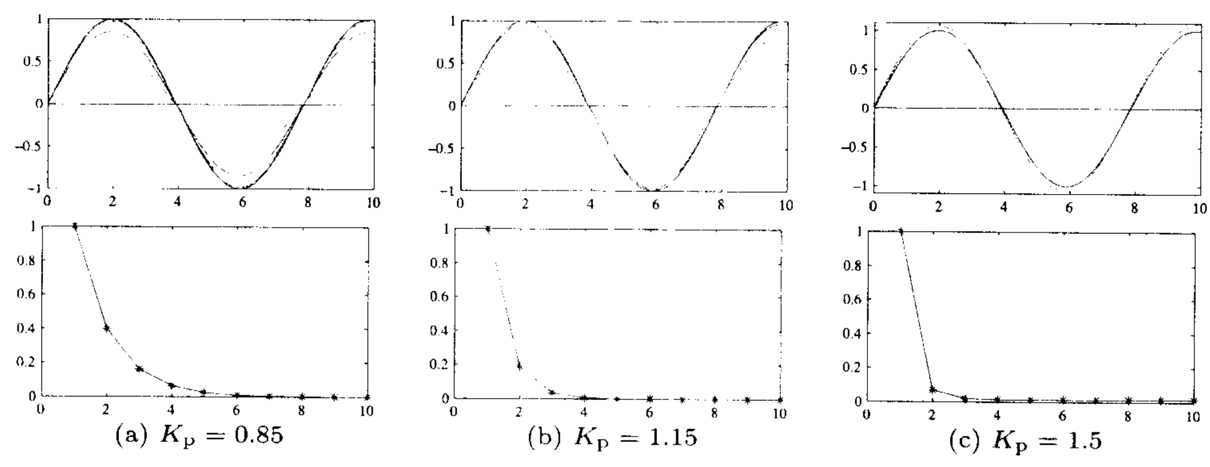


(a) 输出曲线 (b) 跟踪误差范数 (c) 控制信号

图 7-51 迭代学习控制的控制效果

如果采用不同的控制器参数，如 $K_\mathrm{p} = 0.85, 1.15, 1.5$，则可以得出如图 7-52 (a)~(c) 所示的控制结果。可见，这些响应曲线在几次控制后都收敛于期望的 $y_\mathrm{d}(t)$ 曲线，但收敛的速度是不同的。

```
>> [y1,e,u,kv1,t]=ilc_lsim(G,0.1,10,yd,0.85,0,0);
   [y2,e,u,kv2,t]=ilc_lsim(G,0.1,10,yd,1.15,0,0);
   [y3,e,u,kv3,t]=ilc_lsim(G,0.1,10,yd,1.5,0,0);
   subplot(231), plot(t,y1), subplot(234), plot(kv1,'-*')
   subplot(232), plot(t,y2), subplot(235), plot(kv2,'-*')
   subplot(233), plot(t,y3), subplot(236), plot(kv3,'-*')
```



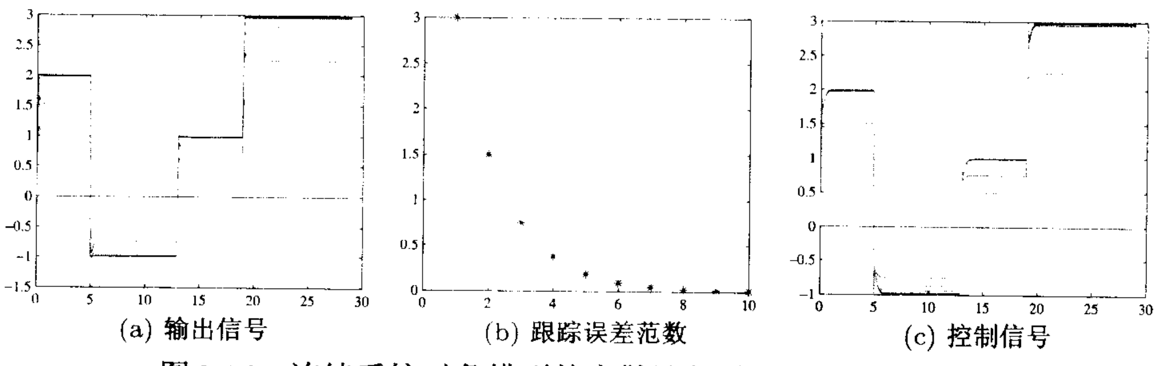
(a) $K_p = 0.85$　　(b) $K_p = 1.15$　　(c) $K_p = 1.5$

图 7-52　不同增益下的迭代学习控制的控制效果

例 7-25　假设某受控对象的数学模型为 $G(s) = \dfrac{0.2s+1}{(0.1s+1)^2}$，试求出系统在迭代学习控制下多阶梯函数输入的控制效果。

求解　用下面的语句可以输入该系统模型，并生成期望的多阶梯输入信号，这样可以通过 ilc_lsim() 函数得出控制效果，如图 7-53 所示。

```
>> s=tf('s'); G=(0.2*s+1)/(0.1*s+1)^2;
   yd=[2*ones(50,1); -1*ones(80,1); 1*ones(60,1); 3*ones(100,1)];
   [y,e,u,kv,t]=ilc_lsim(G,0.1,10,yd,0.5,0,0);
   subplot(131), plot(t,y), subplot(132), plot(t,u),
   subplot(133), plot(kv,'-*'),
```



(a) 输出信号　　(b) 跟踪误差范数　　(c) 控制信号

图 7-53　连续受控对象模型的离散迭代学习控制的控制效果

3. 高阶开环 PID 型 ILC 算法

高阶 ILC 算法是在基本 PID 型 ILC 算法上提出的，高阶 ILC 算法与普通学习算法相比，不仅利用前一次过程的输入输出信息，而且利用了以前多次过程的输入输出信息来构造当前的控制输出．高阶 PID 算法的一般形式为

$$\boldsymbol{u}_k(t)=\sum_{i=1}^{N-1}\left[\boldsymbol{u}_{k-i}(t)+\boldsymbol{\Gamma}_{\mathrm{p}}e_{k-i}(t)+\boldsymbol{\Gamma}_{\mathrm{i}}\int_0^t \boldsymbol{e}_{k-i}(\tau)\mathrm{d}\tau+\boldsymbol{\Gamma}_{\mathrm{d}}\dot{\boldsymbol{e}}_{k-i}(t)\right] \tag{7-4-9}$$

其中，整数 $N\geqslant 2$ 是算法的阶次。

4. 加权 PID 型 ILC 算法和带遗忘因子的 ILC 算法

加权 PID 型 ILC 算法是对以前的过程信息采取不同的权重，从而使以前的过程信息对当前控制作用的影响不同。变增益 ILC 算法也属于加权类算法，同时也具有适应和自整定的意义。引入遗忘因子是为了减少控制输入初始误差的积累对过程的影响，随着迭代次数的增加，越早的控制作用逐渐减小，这样可以使控制信号的变化比较平缓。

5. 其他 ILC 算法

其他 ILC 算法是指除 PID 形式以外的算法。基于高级反馈控制的 ILC 算法有自适应 ILC、预测控制 ILC、最优控制 ILC、智能 ILC、鲁棒 ILC、基于模型的 ILC 等。除此之外，还有针对区间参数系统的 ILC 控制算法[27]等。以上学习算法，有兴趣的读者可查阅相关文献，此处不再赘述。

7.5 习题与思考题

1 考虑例 6-25 中给出的变参数模型，试采用模糊 PD 控制器和模糊 PID 控制对该系统进行仿真研究。

2 对本章例中介绍的模糊 PD 控制器及仿真系统进行研究，若选择很小的 $K_{\mathrm{p}},K_{\mathrm{d}}$ 值将得出什么样的控制效果，为什么？若想获得较好的控制效果，应该如何调整 $K_{\mathrm{p}},K_{\mathrm{d}}$ 甚至 K_{u} 的值。

3 已知表 7-9 中给出的样本点 (x_i,y_i) 数据，试利用神经网络理论在 $x\in(1,10)$ 求解绘制出样本对应的函数曲线。还可以尝试不同的神经网络结构和训练算法，得出较好的拟合效果。

表 7-9 习题 3 数据

x_i	1	2	3	4	5	6	7	8	9	10
y_i	244.0	221.0	208.0	208.0	211.5	216.0	219.0	221.0	221.5	220.0

4 假设已知实测数据由表 7-10 给出，试利用神经网络对 (x,y) 在 $(0.1,0.1) \sim (1.1,1.1)$ 区域内的点进行插值，并用三维曲面的方式绘制出基于神经网络的插值结果。

表 7-10　习题 4 数据

y_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
0.1	0.83041	0.82727	0.82406	0.82098	0.81824	0.8161	0.81481	0.81463	0.81579	0.81853	0.82304
0.2	0.83172	0.83249	0.83584	0.84201	0.85125	0.86376	0.87975	0.89935	0.92263	0.94959	0.9801
0.3	0.83587	0.84345	0.85631	0.87466	0.89867	0.9284	0.96377	1.0045	1.0502	1.1	1.1529
0.4	0.84286	0.86013	0.88537	0.91865	0.95985	1.0086	1.0642	1.1253	1.1904	1.257	1.3222
0.5	0.85268	0.88251	0.92286	0.97346	1.0336	1.1019	1.1764	1.254	1.3308	1.4017	1.4605
0.6	0.86532	0.91049	0.96847	1.0383	1.118	1.2046	1.2937	1.3793	1.4539	1.5086	1.5335
0.7	0.88078	0.94396	1.0217	1.1118	1.2102	1.311	1.4063	1.4859	1.5377	1.5484	1.5052
0.8	0.89904	0.98276	1.082	1.1922	1.3061	1.4138	1.5021	1.5555	1.5573	1.4915	1.346
0.9	0.92006	1.0266	1.1482	1.2768	1.4005	1.5034	1.5661	1.5678	1.4889	1.3156	1.0454
1	0.94381	1.0752	1.2191	1.3624	1.4866	1.5684	1.5821	1.5032	1.315	1.0155	0.62477
1.1	0.97023	1.1279	1.2929	1.4448	1.5564	1.5964	1.5341	1.3473	1.0321	0.61268	0.14763

5 考虑典型线性受控对象模型 $G(s)=\dfrac{\mathrm{e}^{-10s}}{2s+1}$，试采用本章介绍的各种模糊和神经网络控制器对其进行控制，并与 PID 控制比较控制效果。

6 考虑 Rosenbrock 教授提出的最优化问题[28]

$$J=\min_{\boldsymbol{x}\ \text{s.t.}\ -2.048\leqslant x_{1,2}\leqslant 2.048} 100(x_1^2-x_2)+(1-x_1)^2$$

试用遗传算法求解该问题，并与传统最优化方法得出的结果进行比较。

7 De Jong 最优化问题[12]是一个富有挑战性的最优化基准测试问题，其目标函数为

$$J=\min_{\boldsymbol{x}} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}=\min_{\boldsymbol{x}}(x_1^2+x_2^2+\cdots+x_{20}^2)$$

若 $-512\leqslant x_i\leqslant 512, i=1,2,\cdots,20$，试用遗传算法得出其最优化问题的解，并用普通的无约束最优化算法函数 `fminunc()` 求解同样的问题，比较两种方法所需的时间和精度。显然，该问题的全局最优解为 $x_1=x_2=\cdots=x_{20}=0$。

8 试用粒子群算法求解前面的几个问题。

9 试采用遗传算法为下面的受控对象设计最优 PID 控制器。

① 非最小相位系统：$G(s)=\dfrac{-s+5}{s^3+4s^2+5s+6}$

② 不稳定非最小相位系统: $G(s)=\dfrac{-0.2s+5}{s^4+3s^3+5s^2-6s+9}$

③ 不稳定采样系统: $H(z)=\dfrac{4z-2}{z^4+2.9z^3+2.4z^2+1.4z+0.4}$

10 试利用遗传算法求解下面的有约束最优化问题，并和传统数值方法进行比较。

$$\max_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}0.003079x_1^3x_2^3x_5-\cos^3 x_6\geqslant 0\\ 0.1017x_3^3x_4^3-x_5^2\cos^3 x_6\geqslant 0\\ 0.09939(1+x_5)x_1^3x_2^2-\cos^2 x_6\geqslant 0\\ 0.1076(31.5+x_5)x_3^3x_4^2-x_5^2\cos^2 x_6\geqslant 0\\ x_3x_4(x_5+31.5)-x_5[2(x_1+5)\cos x_6+x_1x_2x_5\geqslant 0\\ 0.2\leqslant x_1\leqslant 0.5,14\leqslant x_2\leqslant 22,0.35\leqslant x_3\leqslant 0.6\\ 16\leqslant x_4\leqslant 22.5.8\leqslant x_5\leqslant 6.5,0.14\leqslant x_6\leqslant 0.2618\end{array}\right.} \frac{1}{2\cos x_6}\left[x_1x_2(1+x_5)+x_3x_4\left(1+\frac{31.5}{x_5}\right)\right]$$

11 试利用 MATLAB 语言编写迭代学习控制器的闭环算法和仿真模块。

12 试利用 MATLAB 语言编写高阶 PID 型开环迭代学习控制器。

13 假设控制系统的状态方程[29]为 $\left\{\begin{array}{l}\dot{x}_1(t)=\dfrac{\sin x_1(t)+2\sin x_2(t)}{1+t}+3tu(t)\\ \dot{x}_2(t)=0.3\sin x_1(t)+\dfrac{\sin x_2(t)}{1+t}u(t)\end{array}\right.$，且 $\boldsymbol{x}^{\mathrm{T}}(0)=[0.9,0.9]$，输出方程为 $y(t)=\sin x_(t)+0.5\sin x_2(t)+0.5u(t)$，要求在 $t\in[0,2\pi]$ 时间内跟踪期望输出 $y_{\mathrm{d}}(t)=\sin t$，试构造 P 型闭环迭代学习控制。

参考文献

[1] Fu K S. Learning control systems and intelligent control systems: an intersection of artificial intelligence and automatic control [J]. IEEE Transactions on Automatic Control, 1971, AC-16(1):70~72

[2] 李人厚. 智能控制理论和方法 [M]. 西安: 西安电子科技大学出版社, 1999

[3] 蔡自兴. 智能控制 —— 基础与应用 [M]. 北京: 国防工业出版社, 1998

[4] Zadeh L A. Fuzzy sets [J]. Information and Control, 1965, 8:338~353

[5] 汪培庄. 模糊集合论及其应用 [M]. 上海: 上海科学技术出版社, 1983

[6] The MathWorks Inc. Fuzzy logic toolbox user's manual [Z], 2005

[7] 诸静. 模糊控制原理与应用 [M]. 北京: 机械工业出版社, 1995

[8] 刘金琨. 先进 PID 控制及其 MATLAB 仿真 [M]. 北京: 电子工业出版社, 2003

[9] Hagan M T, Demuth H B, Beale M H. Neural network design [M]. PWS Publishing Company, 1995. (戴葵等译. 神经网络设计. 北京: 机械工业出版社, 2002)

[10] Goldberg D E. Genetic algorithms in search, optimization and machine learning [M]. Addison-Wesley, 1989

[11] 邵军力, 张景, 魏长华. 人工智能基础 [M]. 北京: 电子工业出版社, 2000

[12] Chipperfield A, Fleming P. Genetic algorithm toolbox user's guide [M]. Department of Automatic Control and Systems Engineering, University of Sheffield, 1994

[13] Houck C R, Joines J A, Kay M G. A genetic algorithm for function optimization: a MATLAB implementation [M]. GAOT 工具箱手册电子版, 1995

[14] The MathWorks Inc. Genetic algorithm and direct search toolbox — User's guide 2.0 [Z], 2005

[15] Kennedy J, Eberhart R. Particle swarm optimization [A]. Proceedings of IEEE International Conference on Neural Networks [C]. Perth, Australia, 1995, 1942~1948

[16] Birge B. PSOt, a particle swarm optimization toolbox for MATLAB [A]. Proceedings of the 2003 IEEE Swarm Intelligence Symposium [C]. Indianapolis, 2003, 182~186

[17] Trelea I C. The particle swarm optimization algorithm: convergence analysis and parameter selection [J]. Information Processing Letters, 2003, 85(6):317~325

[18] Clerc M, Kennedy J. The particle swarm: explosion, stability, and convergence in a multidimensional complex space. [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1):58~73

[19] Arimoto S, Kawamura S, Miyazaki F. Bettering operation of robots by learning [J]. Journal of Robotic Systems, 1984, 1:123~140

[20] Moore K L and. Dahleh M, Bhattacharyya S P. Iterative learning control: a survey and new results [J]. Journal of Robotic Systems, 1992, 9(5):563~594

[21] Moore K, Chen Y Q, Ahn H S. Iterative learning control: a tutorial and big picture view [A]. Proceedings of IEEE Conference on Decision and Control [C]. 2006, 2352~2357

[22] Bristow D A, Tharayil M, Alleyne A G. A survey of iterative learning control: A learning-based method for high-performance tracking control [J]. IEEE Control Systems Magazine, 2006, 26(3):96~114

[23] Ahn H S, Chen Y Q, Moore K L. Iterative learning control: brief survey and categorization 1998 ~ 2004 [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part-C, 2006, in press

[24] 谢胜利，田森平，谢振东. 迭代学习控制的理论与应用 [M]. 北京: 科学出版社, 2005

[25] Chen Y Q, Moore K, Yu J, et al. Iterative learning control and repetitive control in hard disk drive industry — a tutorial [A]. Proceedings of IEEE Conference on Decision and Control [C]. 2006

[26] Sugie T, Ono T. An iterative learning control law for dynamical systems [J]. Automatica, 1991, 27(4):729~732

[27] Ahn H-S, Moore K L, Chen Y Q. Iterative learning control — parametric interval robustness and stochastic convergence [M]. Springer-verlag, 2007. To appear

[28] Rosenbrock H H. An automatic method for finding the greatest or least value of a function [J]. Computer Journal, 1960, 3:175~184

[29] 林辉，王林. 迭代学习控制理论 [M]. 西安: 西北工业大学出版社, 1998

第 8 章 鲁棒控制理论的数学基础

经典控制理论中，一般假定系统的受控对象模型已知，这样可采用固定的控制器实现系统的控制。若受控对象模型未知，则需要采用可变的控制器，如模糊逻辑控制器、自校正控制器等对系统进行控制。那么，用固定的控制器能否很好地控制变化的系统模型？鲁棒控制器的出现和发展很好地回答了这样的问题。本章将以这样的问题为背景介绍各种鲁棒控制器的设计方法。

本章 8.1 节给出不确定性的概念与分类、描述方法和不确定边界的提取方法，为不确定性系统的研究奠定基础。8.2 节介绍基于范数指标的鲁棒控制器设计方法和小增益定理，探讨鲁棒控制器的结构，给出鲁棒控制的增广灵敏度的概念与系统增广方法，并介绍基于 MATLAB 鲁棒控制工具箱的 $\mathcal{H}_2$ 最优控制器设计、一般 $\mathcal{H}_\infty$ 控制器设计和最优 $\mathcal{H}_\infty$ 控制器设计方法。8.3 节介绍鲁棒控制器的线性矩阵不等式 (linear matrix inequalities，LMI) 设计方法以及三种形式的 LMI 问题的求解方法，并介绍基于 YALMIP 工具箱的简单求解程序，对比各种不同的鲁棒控制器设计方法和控制效果。8.4 节介绍基于定量反馈理论 (quantitative feedback theory，QFT) 的不确定系统控制器设计方法，给出 QFT 控制器的二自由度控制器结构以及 QFT 控制器设计的步骤，并利用 MATLAB 的 QFT 工具箱详细分析控制器设计的实例。

由于多变量系统不同输入、输出信号间的耦合现象使得控制系统设计很烦琐，所以 8.5 节介绍基于状态反馈的多变量系统解耦算法，根据算法给出解耦控制器的设计方法以及极点配置和标准传递函数相结合的多变量系统状态反馈解耦方法与实现。

8.1 不确定性描述

研究一般鲁棒控制系统，可以用图 8-1 中给出的反馈控制系统表示整个闭环控制系统。其中，$\boldsymbol{P}(s)$ 为受控对象模型，该模型含有不

确定元素，$\boldsymbol{F}(s)$ 为反馈控制器模型。$\boldsymbol{r}(t)$ 和 $\boldsymbol{y}(t)$ 分别为系统的输入和输出信号，$\boldsymbol{d}_{\rm i}(t)$ 和 $\boldsymbol{d}(t)$ 是作用在受控对象输入端和输出端的扰动信号，$\boldsymbol{n}(t)$ 为量测噪声信号。

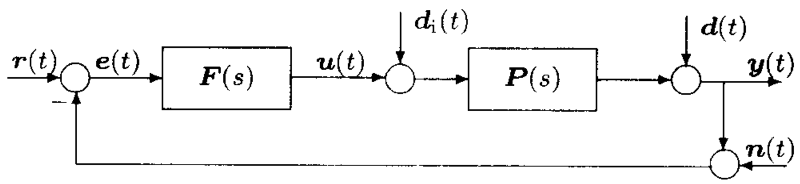
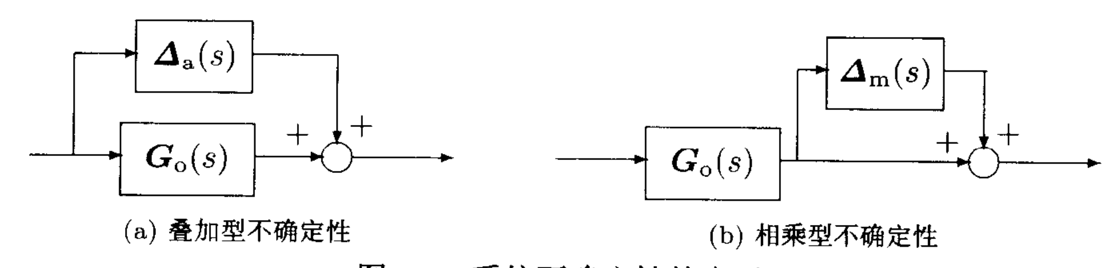


图 8-1 反馈控制系统框图

由这里给出的反馈控制系统模型，可以引入系统的不确定性描述方法和不确定性的 MATLAB 表示。不确定性主要分为内部不确定性和外部不确定性，内部不确定性主要是由系统的模型及其变化产生的，而外部不确定性是由外部扰动信号和噪声信号产生的。下面对不确定性进行深入研究，并探讨不确定性作用的抑制是鲁棒控制需要解决的问题 [1]。

8.1.1 不确定性类型

从建模角度可以将不确定性分成两类，即结构化型不确定性与非结构化型不确定性。结构化型不确定性一般表现对象的动态参数变化，非结构化型不确定性可以进一步地分成叠加型不确定性与相乘型不确定性，这两种不确定性分别如图 8-2 (a) 和 (b) 所示。



(a) 叠加型不确定性 (b) 相乘型不确定性

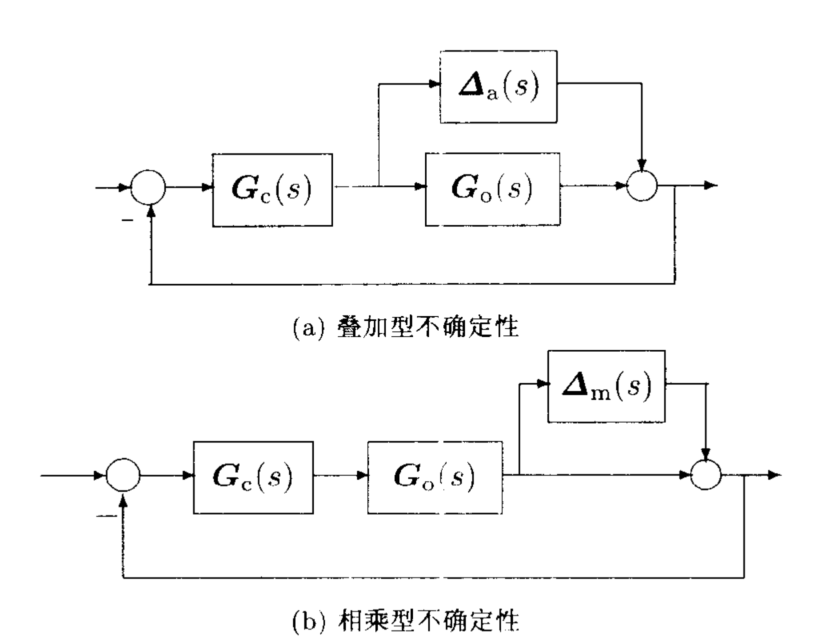
图 8-2 系统不确定性的表示

如果不确定性是叠加型的，则不确定性的模型可以由 $\boldsymbol{\Delta}_{\rm a}(s)=\boldsymbol{G}(s)-\boldsymbol{G}_{\rm o}(s)$ 表示，其中 $\boldsymbol{G}(s)$ 为实际系统模型，而 $\boldsymbol{G}_{\rm o}(s)$ 为系统的标称模型。叠加型的不确定性经常用来表示高频的动态建模误差。

带有叠加型和相乘型不确定性的控制系统结构分别如图 8-3 (a) 和 (b) 所示。 对如图 8-2 (b) 所示的相乘型不确定性来说，其不确定性可以表示成 $\boldsymbol{\Delta}_{\rm m}(s)=[\boldsymbol{G}_{\rm o}(s)-\boldsymbol{G}(s)]\boldsymbol{G}_0^{-1}(s)$。相乘型不确定性模型经常用来表示模型的相对误差，并经常表示模型检测噪声的动态特性。

对叠加型不确定性和相乘型不确定性的传递函数模型 $\boldsymbol{\Delta}_{\rm a}(s)$ 和 $\boldsymbol{\Delta}_{\rm m}(s)$ 来说，若下面的不等式

- **叠加型不确定性：** $|\boldsymbol{\Delta}_{\rm a}({\rm j}\omega)|<\dfrac{1}{|\boldsymbol{G}_{\rm c}({\rm j}\omega)\boldsymbol{S}({\rm j}\omega)|}=\boldsymbol{B}_{\rm a}({\rm j}\omega)$



(a) 叠加型不确定性

(b) 相乘型不确定性

图 8-3　带有不确定性的控制系统结构

- **相乘型不确定性**: $|\boldsymbol{\Delta}_{\rm m}({\rm j}\omega)| < \dfrac{1}{|\boldsymbol{T}({\rm j}\omega)|} = \boldsymbol{B}_{\rm m}({\rm j}\omega)$

对所有的频率 ω 都成立，则称闭环系统鲁棒稳定。其中，$\boldsymbol{S}({\rm j}\omega)$ 和 $\boldsymbol{T}({\rm j}\omega)$ 称为系统的灵敏度和补灵敏度函数，定义为

$$\boldsymbol{S}({\rm j}\omega) = [\boldsymbol{I} + \boldsymbol{G}({\rm j}\omega)\boldsymbol{F}({\rm j}\omega)]^{-1},\ \ \boldsymbol{T}({\rm j}\omega) = \boldsymbol{I} - \boldsymbol{S}({\rm j}\omega) \tag{8-1-1}$$

$\boldsymbol{B}_{\rm a}({\rm j}\omega)$ 和 $\boldsymbol{B}_{\rm m}({\rm j}\omega)$ 的值经常又称作扰动信号的边界，这两个量和频率有关。

8.1.2　不确定性描述方法

鲁棒控制工具箱定义了一个新的对象类 `ureal`，用其可以给定在某个区间内可变的变量，该函数的调用格式为

```
p=ureal('p',p0,'Range',[pm,pM])          % p ∈ [pm,pM]
p=ureal('p',p0,'PlusMinus',δ)            % p ∈ [p0 − δ,p0 + δ]
p=ureal('p',p0,'Percentage',A)           % p ∈ [p0 − A%,p0 + A%]
```

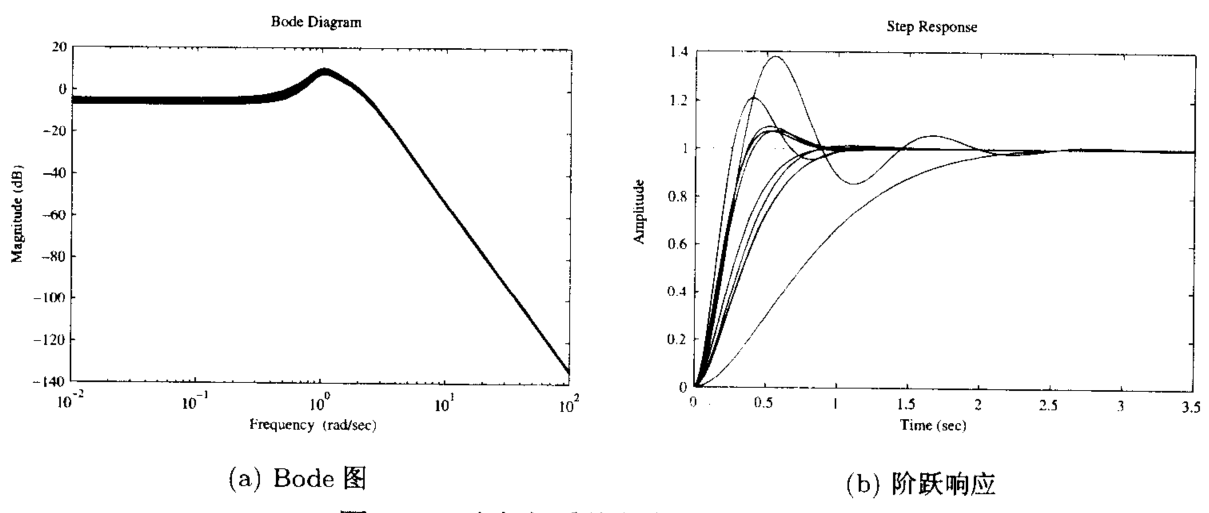
其中 p_0 为该变量的标称值，其变化范围可以由后面的参数直接定义。有了这样的不确定变量，则可以由 `tf()` 或 `ss()` 函数建立起不确定系统的传递函数或状态方程模型。有了数学模型，还可以用 G_1=`usample(`G,N`)` 函数，采用 Monte Carlo 方法从不确定系统 G 中随机选择 N 个样本赋给 G_1。

例 8-1　考虑不确定动态系统[2]

$$G(s,a,b) = \frac{10[(2+0.2a)s^2 + (2+0.3a+0.4)s + (1+0.2b)]}{(s^2+0.5s+1)(s^2+2s+3)(s^2+3s+6)},\ \ (a,b) \in (-1,1)$$

选定标称值为 $a=b=0$。试将该不确定系统输入 MATLAB 工作空间，并绘制出不确定系统的开环 Bode 图与闭环阶跃响应曲线，并研究叠加型不确定性和相乘性不确定性的频域曲线。

求解　显然，由前面介绍的方法先定义两个不确定变量，然后就可以构造出不确定系统模型了。从构造的模型中随机选择 15 个样本，则可以由简单、直观的命令绘制出样本系统的开环 Bode 图和闭环阶跃响应曲线，如图 8-4 所示。值得说明的是，每次调用 usample() 函数得出的样本将是不同的。



(a) Bode 图　　(b) 阶跃响应

图 8-4　不确定系统的频域与时域分析

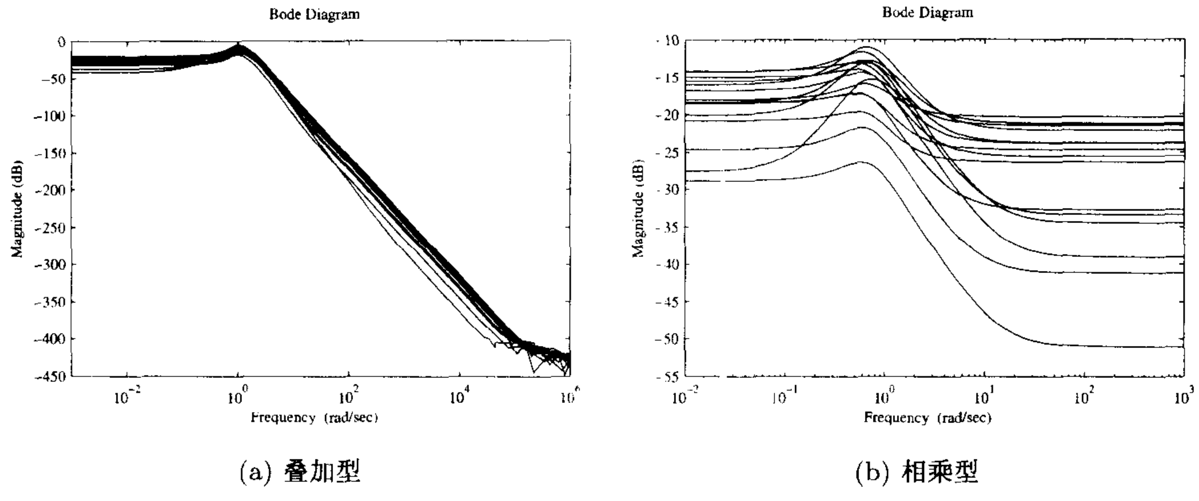
```
>> a=ureal('a',0,'Range',[-1,1]); b=ureal('b',0,'Range',[-1,1]);
   den=conv(conv([1 0.5 1],[1 2 3]),[1 3 6]);
   G=10*tf([2+0.2*a,2+0.3*a+0.4*b,1+0.2*b],den);
   P0=10*tf([2,2,1],den); P=usample(G,15);  bodemag(P);
   figure; G1=feedback(G,1); P1=usample(G1,15); step(P1)
```

考虑叠加型和相乘型不确定性，下面的语句可以分别绘制出不确定性的 Bode 图，如图 8-5 (a) 和 (b) 所示。

```
>> DelA=P-P0; bodemag(DelA), figure; DelM=(P0-P)/P0; bodemag(DelM)
```

8.1.3　摄动边界函数的提取与建模

图 8-5 中得出了因参数变化导致的不确定性的 Bode 幅值响应曲线，从这样得出的不确定性频率响应可见，若选择其包络线或任何单一的频域响应曲线，使得在某一频率下增益均高于不确定性的增益，则可以将不确定性的复杂模型用一个单一的传递函数 $W_{\rm a}(s)$ 或 $W_{\rm m}(s)$ 来表示，对于相加型或相乘型不确定性满足 $|\Delta_{\rm a}({\rm j}\omega)|\leqslant W_{\rm a}(s)$ 或 $|\Delta_{\rm m}({\rm j}\omega)|\leqslant W_{\rm m}(s)$，则称 $W_{\rm a}(s)$ 或 $W_{\rm m}(s)$ 为摄动边界函



(a) 叠加型 (b) 相乘型

图 8-5 不确定性的 Bode 图

数。从鲁棒控制的角度看，如果能找到使得处于边界的系统稳定，则可以使得任意处于边界以内的系统稳定，所以摄动边界函数在鲁棒控制器设计中是很重要的概念。

选择增益的边界函数应该考虑几个因素：

① **减小保守性** 如果将增益的边界函数选择得比实际可能发生的增益高太多将带来很大的保守性，这可能过分地牺牲控制器的动态性能，所以从减少保守性角度看，有时需考虑采用接近可能增益的包络线作为边界函数。

② **低阶函数逼近** 通常情况下，如果能采用低阶模型逼近边界函数，将降低控制器的阶次。另外，用简单折线的形式描述低阶模型，比较方便。

文献 [2] 给出了提取扰动边界的方法，首先绘制出参数变化时系统频域响应的幅频特性，用 `ginput()` 函数在边界上读取一些点，然后用 `fitmag()` 函数找出边界的传递函数模型。当然，还可以编写一个小程序提取上界的包络线，加一个偏差，留有余量，这样就可以识别其传递函数了。

根据上面的思路可以编写如下的函数来求取包络线

```
function W=up_evolope(Gerr,w,n)
H=frd(Gerr,w); H1=H.Response(:); M=length(w);
H1=reshape(abs(H1),M,length(H1)/M);
H1=max(abs(H1)'); H1=frd(H1,w); W=fitmagfrd(H1,n);
```

该函数的调用格式为 W=`up_envolop(`G,w,n`)`，其中 G 为误差模型，w 为频率向量，n 为预期的阶次，W 为包络线的拟合状态方程模型。

例 8-2 考虑例 8-1 中给出的模型，试求出相乘型不确定性的包络线模型。

求解 由下面的语句可以得出包络线对应的三阶模型，得出的 Bode 图如图 8-6 所示（粗线表示包络线模型）。可见，这样得出的模型拟合效果比较理想。

```
>> a=ureal('a',0,'Range',[-1,1]); b=ureal('b',0,'Range',[-1,1]);
   den=conv(conv([1 0.5 1],[1 2 3]),[1 3 6]);
   G=10*tf([2+0.2*a,2+0.3*a+0.4*b,1+0.2*b],den); DelM=(P0-P)/P0;
   w=logspace(-1,2); W=zpk(up_envolop(DelM,w,3))
   bodemag(DelM,'b',W,'r')
```

这样得出的包络线传递函数为 $W(s)=\dfrac{0.097328(s+0.48)(s^2+6.141s+11.28)}{(s+4.414)(s^2+1.084s+0.6142)}$。这样，不确定模型的误差可以写成 $|\Delta_{\rm m}(j\omega)|\leqslant|W(j\omega)|$。

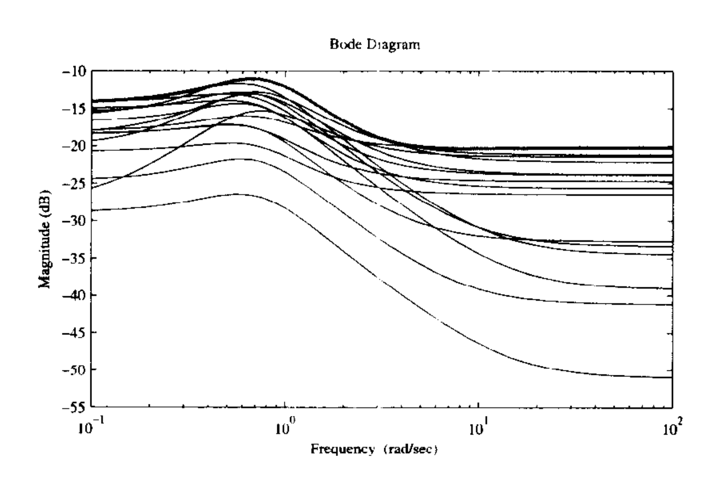


图 8-6 不确定性的 Bode 图及包络线模型的 Bode 图

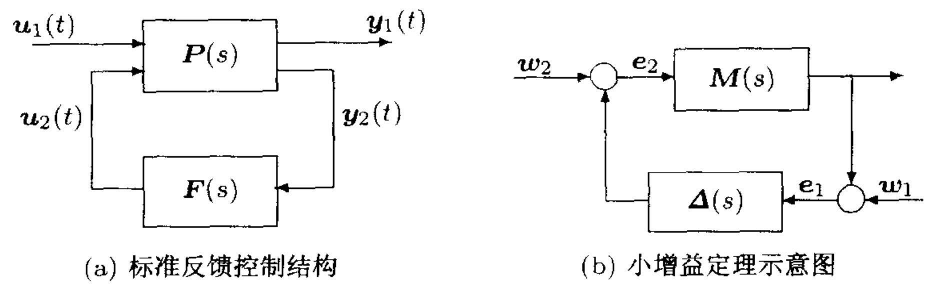
8.2 基于范数的鲁棒控制理论

基于范数的鲁棒控制理论最早是由 George Zames 教授提出的[3]，当时主要讨论灵敏度函数的 $\mathcal{H}_\infty$ 范数，控制目标是使得受扰动系统在扰动边界处的最坏情况下，仍然能设计出满足要求的控制器。这种思想开创了一个很重要的控制分支的出现与发展。本节首先介绍鲁棒控制的一个重要的理论基础 —— 小增益定理，然后介绍基于范数的控制系统结构与设计。

8.2.1 小增益定理

鲁棒控制系统的一般结构如图 8-7 (a) 所示，其中 $\boldsymbol{P}(s)$ 为增广的对象模型，而 $\boldsymbol{F}(s)$ 为控制器模型。从输入信号 $\boldsymbol{u}_1(t)$ 到输出信号 $\boldsymbol{y}_1(t)$ 的传递函数可以表示为 $\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(t)$。在鲁棒控制中，小增益定理是个很关键的理论基础，下面将叙述这个定理。

假设 $\boldsymbol{M}(s)$ 为稳定的，则当且仅当小增益条件 $||\boldsymbol{M}(s)||_\infty||\boldsymbol{\Delta}(s)||_\infty<1$ 满足时，图 8-7 (b) 中所示的系统对所有稳定的 $\boldsymbol{\Delta}(s)$ 都是良定的，且是内部稳定的。



(a) 标准反馈控制结构　　(b) 小增益定理示意图

图 8-7 $\mathcal{H}_2$ 与 $\mathcal{H}_\infty$ 控制的一般结构

事实上，可以这样理解小增益定理：如果系统的回路传递函数的范数小于 1，则闭环系统将总是稳定的。

8.2.2 鲁棒控制器的结构

在如图 8-7(a) 所示的闭环系统结构中，引入了增广的对象模型，该模型一般可以表示成

$$\boldsymbol{P}(s)=\begin{bmatrix}\boldsymbol{P}_{11}(s) & \boldsymbol{P}_{12}(s)\\ \boldsymbol{P}_{21}(s) & \boldsymbol{P}_{22}(s)\end{bmatrix}=\left[\begin{array}{c|cc}\boldsymbol{A} & \boldsymbol{B}_1 & \boldsymbol{B}_2\\ \hline \boldsymbol{C}_1 & \boldsymbol{D}_{11} & \boldsymbol{D}_{12}\\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{D}_{22}\end{array}\right] \tag{8-2-1}$$

其对应的增广状态方程描述为

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}+[\boldsymbol{B}_1\ \boldsymbol{B}_2]\begin{bmatrix}\boldsymbol{u}_1\\ \boldsymbol{u}_2\end{bmatrix},\quad \begin{bmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_2\end{bmatrix}=\begin{bmatrix}\boldsymbol{C}_1\\ \boldsymbol{C}_2\end{bmatrix}\boldsymbol{x}+\begin{bmatrix}\boldsymbol{D}_{11} & \boldsymbol{D}_{12}\\ \boldsymbol{D}_{21} & \boldsymbol{D}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_1\\ \boldsymbol{u}_2\end{bmatrix} \tag{8-2-2}$$

闭环系统的框图可以绘制成如图 8-8 所示的形式[4]，其闭环系统传递函数可以写成

$$\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)=\boldsymbol{P}_{11}(s)+\boldsymbol{P}_{12}(s)\Big[\boldsymbol{I}-\boldsymbol{F}(s)\boldsymbol{P}_{22}(s)\Big]^{-1}\boldsymbol{F}(s)\boldsymbol{P}_{21}(s) \tag{8-2-3}$$

这样的结构在控制理论中常常称为线性分式变换。鲁棒控制的目的是设计出一个镇定控制器 $\boldsymbol{u}_2(s)=\boldsymbol{F}(s)\boldsymbol{y}_2(s)$，使得闭环系统 $\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)$ 的范数取一个小于 1 的值，亦即 $||\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)||<1$。从式 (8-2-3) 出发，常常可以将鲁棒控制问题分为下面三种形式：

① $\mathcal{H}_2$ **最优控制问题** 其中需求解 $\min||\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)||_2$；

② $\mathcal{H}_\infty$ **最优控制问题** 其中需求解 $\min||\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)||_\infty$；

③ **标准 $\mathcal{H}_\infty$ 控制问题** 需要得出一个控制器满足 $||\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)||_\infty<1$。

加权的控制结构如图 8-9 (a) 所示，其中 $\boldsymbol{W}_1(s)$, $\boldsymbol{W}_2(s)$ 与 $\boldsymbol{W}_3(s)$ 都是加权函数，这些加权函数应该使得 $\boldsymbol{G}(s)$，$\boldsymbol{W}_1(s)$ 与 $\boldsymbol{W}_3(s)\boldsymbol{G}(s)$ 均正则。换句话说，这些传递函数在 $s\to\infty$ 时均应该是有界的。可以看出，在这个条件下并没有直接要求 $\boldsymbol{W}_3(s)$ 本身是正则的。对图 8-9 (a) 中的方框图结构稍加改动，则可以容易地

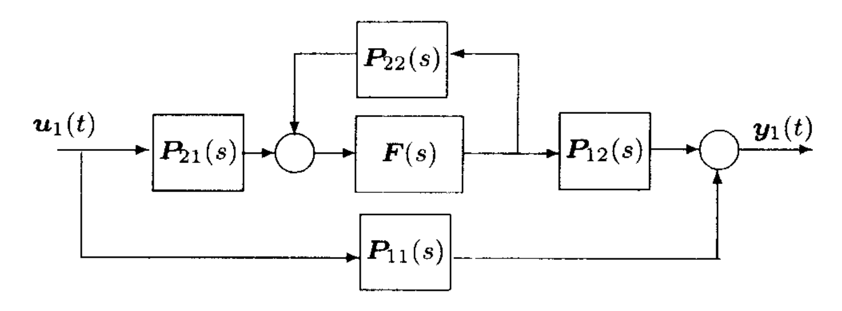


图 8-8 闭环系统框图的另外一种描述方法

得出如图 8-9 (b) 中所示的控制结构，可以看出这样的结构和图 8-7 (a) 中给出的标准鲁棒控制结构是完全一致的。

假定系统对象模型的状态方程为 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$，则加权函数 $\boldsymbol{W}_1(s)$ 的状态方程模型为 $(\boldsymbol{A}_{\mathrm{w}_1},\boldsymbol{B}_{\mathrm{w}_1},\boldsymbol{C}_{\mathrm{w}_1},\boldsymbol{D}_{\mathrm{w}_1})$，$\boldsymbol{W}_2(s)$ 的状态方程模型为 $(\boldsymbol{A}_{\mathrm{w}_2},\boldsymbol{B}_{\mathrm{w}_2},\boldsymbol{C}_{\mathrm{w}_2},\boldsymbol{D}_{\mathrm{w}_2})$，而可以为非正则的 $\boldsymbol{W}_3(s)$ 的模型表示为

$$\boldsymbol{W}_3(s)=\boldsymbol{C}_{\mathrm{w}_3}(s\boldsymbol{I}-\boldsymbol{A}_{\mathrm{w}_3})^{-1}\boldsymbol{B}_{\mathrm{w}_3}+\boldsymbol{P}_m s^m+\cdots+\boldsymbol{P}_1 s+\boldsymbol{P}_0 \tag{8-2-4}$$

特别地，式 (8-2-1) 可以写成

$$\boldsymbol{P}(s)=\left[\begin{array}{cccc:c:c}\boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}\\ -\boldsymbol{B}_{\mathrm{w}_1}\boldsymbol{C} & \boldsymbol{A}_{\mathrm{w}_1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_{\mathrm{w}_1} & -\boldsymbol{B}_{\mathrm{w}_1}\boldsymbol{D}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\mathrm{w}_2} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{B}_{\mathrm{w}_2}\\ \boldsymbol{B}_{\mathrm{w}_3}\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\mathrm{w}_3} & \boldsymbol{0} & \boldsymbol{B}_{\mathrm{w}_3}\boldsymbol{D}\\ \hdashline -\boldsymbol{D}_{\mathrm{w}_1}\boldsymbol{C} & \boldsymbol{C}_{\mathrm{w}_1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{\mathrm{w}_1} & -\boldsymbol{D}_{\mathrm{w}_1}\boldsymbol{D}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{C}_{\mathrm{w}_2} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{\mathrm{w}_2}\\ \widetilde{\boldsymbol{C}}+\boldsymbol{S}_{\mathrm{w}_3}\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{C}_{\mathrm{w}_3} & \boldsymbol{0} & \widetilde{\boldsymbol{D}}+\boldsymbol{D}_{\mathrm{w}_3}\boldsymbol{D}\\ \hdashline -\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & -\boldsymbol{D}\end{array}\right] \tag{8-2-5}$$

式中

$$\begin{aligned}\widetilde{\boldsymbol{C}}&=\boldsymbol{P}_0\boldsymbol{C}+\boldsymbol{P}_1\boldsymbol{C}\boldsymbol{A}+\cdots+\boldsymbol{P}_m\boldsymbol{C}\boldsymbol{A}^{m-1}\\ \widetilde{\boldsymbol{D}}&=\boldsymbol{P}_0\boldsymbol{D}+\boldsymbol{P}_1\boldsymbol{C}\boldsymbol{B}+\cdots+\boldsymbol{P}_m\boldsymbol{C}\boldsymbol{A}^{m-2}\boldsymbol{B}\end{aligned} \tag{8-2-6}$$

其中任何一个加权函数均可以是空的，在 MATLAB 下可以表示为 $\boldsymbol{W}_i(s)=[]$。这时鲁棒控制问题可以集中成下面三种形式来研究：

① **灵敏度问题** 在灵敏度问题中并不指定 $\boldsymbol{W}_2(s)$ 与 $\boldsymbol{W}_3(s)$；

② **稳定性与品质的混合鲁棒问题** 在这样的问题中假定 $\boldsymbol{W}_3(s)$ 为空的；

③ **一般的混合灵敏度问题** 其中要求三个加权函数都存在。

一般情况下的增广对象模型可以写成

$$\boldsymbol{P}(s)=\left[\begin{array}{c:c}\boldsymbol{W}_1 & -\boldsymbol{W}_1\boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{W}_2\\ \boldsymbol{0} & \boldsymbol{W}_3\boldsymbol{G}\\ \hdashline \boldsymbol{I} & -\boldsymbol{G}\end{array}\right] \tag{8-2-7}$$

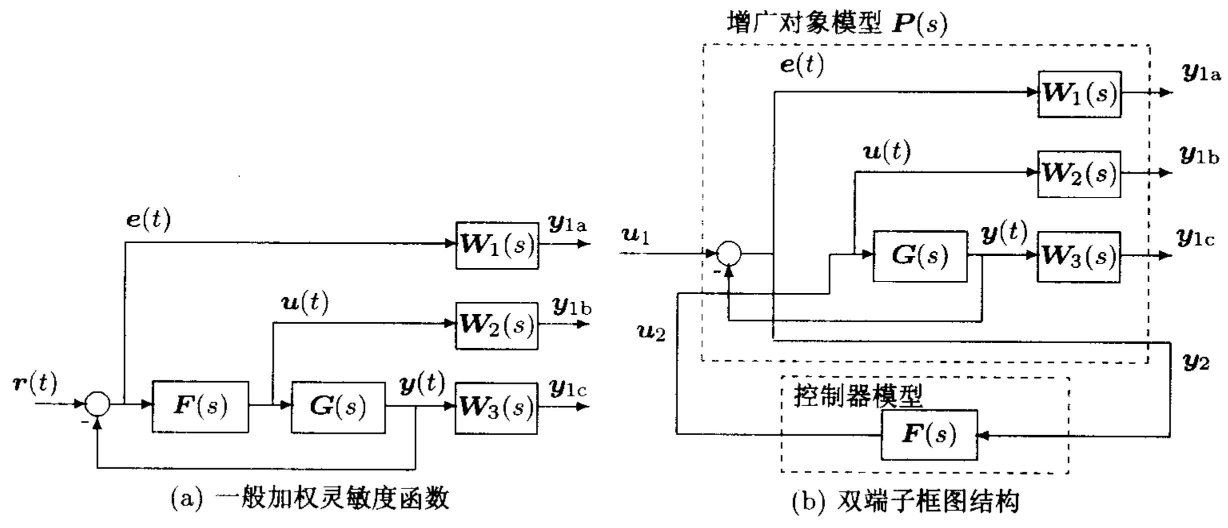


图 8-9　加权灵敏度问题的框图表示

这个结构又称为 $\mathcal{H}_\infty$ 设计的一般混合灵敏度问题。在这样的问题下，线性分式表示可以写成 $\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)=\left[\boldsymbol{W}_1\boldsymbol{S},\boldsymbol{W}_2\boldsymbol{F}\boldsymbol{S},\boldsymbol{W}_3\boldsymbol{T}\right]^{\mathrm{T}}$，其中 $\boldsymbol{F}(s)$ 为控制器模型，$\boldsymbol{S}(s)$ 为灵敏度函数，其定义为 $\boldsymbol{S}(s)=\boldsymbol{E}(s)\boldsymbol{R}^{-1}(s)=[\boldsymbol{I}+\boldsymbol{F}(s)\boldsymbol{G}(s)]^{-1}$，而 $\boldsymbol{T}(s)$ 为补灵敏度函数，其定义为 $\boldsymbol{T}(s)=\boldsymbol{I}-\boldsymbol{S}(s)$。灵敏度是决定跟踪误差大小的最重要指标，灵敏度越低，则系统的跟踪误差越小，故系统响应的品质指标越好，而补灵敏度函数是决定系统鲁棒稳定性的重要指标，它制约着系统输出信号的大小，在存在不确定性时，有较大的加权会迫使系统输出信号稳定[5]。灵敏度和补灵敏度函数的加权选择是相互矛盾的，故它们之间应该存在折中，所以可以认为鲁棒控制器设计是加权函数选取的艺术[6]。

类似于线性时不变系统模型的输入方法，受控对象模型、加权函数模型都可以采用 `ss()` 函数、`tf()` 函数和 `zpk()` 函数直接输入。假设它们的模型分别为 $\boldsymbol{G}$, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ 和 $\boldsymbol{W}_3$，其中 $\boldsymbol{W}_3$ 允许为分子阶次高于分母阶次的模型，则可以由 `augtf()` 函数增广该模型，得出增广系统模型 $\boldsymbol{P}$=augtf($\boldsymbol{G}$,$\boldsymbol{W}_1$,$\boldsymbol{W}_2$,$\boldsymbol{W}_3$)。鲁棒控制工具箱新版本还提供一个统一的 `augw()`，但该函数是存在局限性的，因为它要求所有这些子模型都是正实的，即分子的阶次不能高于分母的阶次。

由 `branch()` 函数可以提取出增广状态方程的各个子矩阵

[$\boldsymbol{A}$,$\boldsymbol{B}_1$,$\boldsymbol{B}_2$,$\boldsymbol{C}_1$,$\boldsymbol{C}_2$,$\boldsymbol{D}_{11}$,$\boldsymbol{D}_{12}$,$\boldsymbol{D}_{21}$,$\boldsymbol{D}_{22}$]=branch($\boldsymbol{P}$)

例 8-3 考虑下面给出的系统状态方程模型

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0 & 1 & 0 & 0\\ -5000 & -100/3 & 500 & 100/3\\ 0 & -1 & 0 & 1\\ 0 & 100/3 & -4 & -60\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\ 25/3\\ 0\\ -1\end{bmatrix}u(t),\quad \boldsymbol{y}(t)=[0,0,1,0]\boldsymbol{x}(t)$$

若选择加权函数 $W_1(s)=100/(s+1), W_3(s)=s/1000$，试得出系统的增广模型。

求解 可以由下面的 MATLAB 命令来建立起增广的对象模型

```
>> A=[0,1,0,0; -5000,-100/3,500,100/3; 0,-1,0,1; 0,100/3,-4,-60];
   B=[0; 25/3; 0; -1]; C=[0,0,1,0]; D=0; G=ss(A,B,C,D); s=tf('s');
   W1=100/(s+1); W2=1e-5; W3=s/1000; % 定义加权函数
   T_ss=augtf(G,W1,W2,W3);  % 得出增广的双端子系统模型
   [a,b1,b2,c1,c2,d11,d12,d21,d22]=branch(T_ss)
```

注意，由于没有 $W_2(s)$ 加权函数，所以应该将其设置成小的正数，如 10^{-5}，以避免式 (8-2-5) 中的 D_{12} 矩阵成为奇异矩阵，导致原问题无解。由下面的 branch() 函数调用命令可以直接提取出增广系统的双端子状态方程模型为

$$
P(s)=\left[\begin{array}{ccccc|c|c}
0 & 1 & 0 & 0 & 0 & 0 & 0\\
-5000 & -33.333 & 500 & 33.333 & 0 & 0 & 8.3333\\
0 & -1 & 0 & 1 & 0 & 0 & 0\\
0 & 33.333 & -4 & -60 & 0 & 0 & -1\\
0 & 0 & -1 & 0 & -1 & 1 & 0\\
\hline
0 & 0 & 0 & 0 & 100 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 10^{-5}\\
0 & -0.001 & 0 & 0.001 & 0 & 0 & 0\\
\hline
0 & 0 & -1 & 0 & 0 & 1 & 0
\end{array}\right]
$$

注意，若 $W_2(s)$ 取成空矩阵时，相应的 D_{12} 矩阵项为 0，导致 D_{12} 矩阵奇异，使得 $\mathcal{H}_\infty$ 设计问题不能求解。在实际系统设计时应该避免这样的问题。

8.2.3 控制系统状态反馈与输出反馈闭环模型

对应前面给出的增广系统模型 $P(s)$，相应的数学模型重新表示为

$$
\begin{cases}
\dot{x}(t)=Ax(t)+B_1u_1(t)+B_2u_2(t)\\
y_1(t)=C_1x(t)+D_{11}u_1(t)+D_{12}u_2(t)\\
y_2(t)=C_2x(t)+D_{21}u_1(t)+D_{22}u_2(t)
\end{cases}
\tag{8-2-8}
$$

若引入状态反馈 $u_2(t)=Kx(t)$ 则可以构造出闭环系统为

$$
\begin{cases}
\dot{x}(t)=(A+B_2K)x(t)+B_1u_1(t)\\
y_1(t)=(C_1+D_{12}K)x(t)+D_{11}u_1(t)
\end{cases}
\tag{8-2-9}
$$

如果引入输出反馈 $u_2(t)=K(s)y_2(t)$，其中控制器的动态模型为

$$
\begin{cases}
\dot{\hat{x}}(t)=A_K\hat{x}(t)+B_Ky(t)\\
u(t)=C_K\hat{x}(t)+D_Ky(t)
\end{cases}
\tag{8-2-10}
$$

这时的闭环系统的状态方程模型为

$$\begin{cases} \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\hat{\boldsymbol{x}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{D}_{\rm K}\boldsymbol{C}_2 & \boldsymbol{B}_2\boldsymbol{C}_{\rm K} \\ \boldsymbol{B}_{\rm K}\boldsymbol{C}_2 & \boldsymbol{A}_{\rm K} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_1+\boldsymbol{B}_2\boldsymbol{D}_{\rm K}\boldsymbol{D}_{21} \\ \boldsymbol{B}_{\rm K}\boldsymbol{D}_{21} \end{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{y}_1 = [\boldsymbol{C}_1+\boldsymbol{D}_{12}\boldsymbol{D}_{\rm K}\boldsymbol{C}_2,\ \boldsymbol{D}_{12}\boldsymbol{C}_{\rm K}] \begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix} + (\boldsymbol{D}_{11}+\boldsymbol{D}_{12}\boldsymbol{D}_{\rm K}\boldsymbol{D}_{21})\boldsymbol{u}_1 \end{cases} \tag{8-2-11}$$

8.2.4 基于鲁棒控制工具箱的设计方法

考虑图 8-7 (a) 中所示的双端子状态方程对象模型结构，$\mathcal{H}_\infty$ 的设计目标是找到一个控制器 $\boldsymbol{F}(s)$，它能保证闭环系统的 $\mathcal{H}_\infty$ 范数限制在一个给定的小正数 γ 下，即 $||\boldsymbol{T}_{\boldsymbol{y}_1\boldsymbol{u}_1}(s)||_\infty < \gamma$。文献 [7] 给出了一种求解方法

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_{\rm f}\boldsymbol{x}(t) - \boldsymbol{Z}\boldsymbol{L}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{K}\boldsymbol{x}(t) \tag{8-2-12}$$

其中，若 $\boldsymbol{D}_{11} = \boldsymbol{0}$，则

$$\begin{aligned} &\boldsymbol{A}_{\rm f} = \boldsymbol{A} + \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^{\rm T}\boldsymbol{X} + \boldsymbol{B}_2\boldsymbol{K} + \boldsymbol{Z}\boldsymbol{L}\boldsymbol{C}_2 \\ &\boldsymbol{K} = -\boldsymbol{B}_2^{\rm T}\boldsymbol{X}, \quad \boldsymbol{L} = -\boldsymbol{Y}\boldsymbol{C}_2^{\rm T}, \quad \boldsymbol{Z} = (\boldsymbol{I} - \gamma^{-2}\boldsymbol{Y}\boldsymbol{X})^{-1} \end{aligned} \tag{8-2-13}$$

且 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 分别为下面两个代数 Riccati 方程的解

$$\begin{aligned} &\boldsymbol{A}^{\rm T}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} - \boldsymbol{X}(\boldsymbol{B}_2\boldsymbol{B}_2^{\rm T} - \gamma^{-2}\boldsymbol{B}_1\boldsymbol{B}_1^{\rm T})\boldsymbol{X} + \boldsymbol{C}_1^{\rm T}\boldsymbol{C}_1 = \boldsymbol{0} \\ &\boldsymbol{A}\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{A}^{\rm T} - \boldsymbol{Y}(\boldsymbol{C}_2^{\rm T}\boldsymbol{C}_2 - \gamma^{-2}\boldsymbol{C}_1^{\rm T}\boldsymbol{C}_1)\boldsymbol{Y} + \boldsymbol{B}_1\boldsymbol{B}_1^{\rm T} = \boldsymbol{0} \end{aligned} \tag{8-2-14}$$

$\mathcal{H}_\infty$ 控制器存在的前提条件为：

① $\boldsymbol{D}_{11}$ 足够小，且满足 $\boldsymbol{D}_{11} < \gamma$；

② 控制器 Riccati 方程的解 $\boldsymbol{X}$ 为非负定矩阵；

③ 观测器 Riccati 方程的解 $\boldsymbol{Y}$ 为非负定矩阵；

④ $\lambda_{\max}(\boldsymbol{X}\boldsymbol{Y}) < \gamma^2$，即两个 Riccati 方程积矩阵的所有特征值均小于 γ^2。

例 8-4 假设已知系统增广模型的各个矩阵如下，试设计 $\mathcal{H}_\infty$ 控制器。

$$\boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},\ \boldsymbol{B}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \\ 0 & 0 \end{bmatrix},\ \boldsymbol{B}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ \boldsymbol{C}_1 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{C}_2 = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix},\ \boldsymbol{D}_{11} = \boldsymbol{D}_{22} = \boldsymbol{0},\ \boldsymbol{D}_{12} = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix},\ \boldsymbol{D}_{21} = [1, 0]$$

求解 可以由底层语句求解 $\mathcal{H}_\infty$ 控制器

```
>> A=[-1,0,0; 0,0,0; 0,1,0]; B1=[1,0; 0,0.01; 0,0]; B2=[0; 1; 0];
   C1=[1,0,-1; 0,0,0]; C2=[0,0,-1]; D11=zeros(2); D12=[0; 0.01];
   D21=[1,0]; D22=0; gam=2;
   X=are(A,B2*B2'-gam^(-2)*B1*B1',C1'*C1);
   Y=are(A',C2'*C2-gam^(-2)*C1'*C1,B1*B1');
   K=-B2'*X; L=-Y*C2'; Z=inv(eye(size(A))-gam^(-2)*Y*X);
   Ar=A+gam^(-2)*B1*B1'*X+B2*K+Z*L*C2; Gc=ss(Ar,-Z*L,K,0)
```

这样得出的 $\mathcal{H}_\infty$ 控制器为

$$G_c(s)=\left[\begin{array}{ccc:c} -0.88056 & -0.079676 & -0.13209 & 0.055202 \\ 0.3187 & -1.4701 & -1.0813 & -0.013445 \\ 0 & 1 & -0.20964 & -0.20964 \\ \hdashline 0.3187 & -1.4701 & -1.0679 & 0 \end{array}\right]$$

根据前面介绍的 $\mathcal{H}_\infty$ 控制器的设计算法和控制器可解的条件，可以编写出下面的 MATLAB 函数，其中将 γ 的值设置成目标函数，如果控制器的条件不满足，则将目标函数的值设置成较大的值，如 10。这样，通过前面介绍的寻优程序就可以计算出最小的 γ 值了。

```
function y=find_hinf(gam,A,B1,B2,C1,C2,D11,D12,D21,D22)
try
   X=are(A,B2*B2'-gam^(-2)*B1*B1',C1'*C1);
   Y=are(A',C2'*C2-gam^(-2)*C1'*C1,B1*B1');
   Z=inv(eye(size(A))-gam^(-2)*Y*X); y=gam;
   if any(eig(X)<0)|any(eig(Y)<0)|max(eig(X*Y))>gam^2, y=10; end
catch, y=10; end
```

例 8-5 重新考虑前面的问题，若给出下面的语句，则可以通过寻优的方法得出最小的 γ 值为 1.4401。

```
>> A=[-1,0,0; 0,0,0; 0,1,0]; B1=[1,0; 0,0.01; 0,0]; B2=[0; 1; 0];
   C1=[1,0,-1; 0,0,0]; C2=[0,0,-1]; D11=zeros(2); D12=[0; 0.01];
   D21=[1,0]; D22=0; x0=5;
   gam=fminsearch(@find_hinf,x0,[],A,B1,B2,C1,C2,D11,D12,D21,D22)
```

值得指出的是，上述的算法并非万能的，在求解实际问题时应该采用鲁棒控制工具箱提供的现成函数。

定义了系统的双端子模型后，前面介绍的三种鲁棒控制器的设计就可以调用鲁棒控制工具箱中相应的函数直接实现了，这些设计函数可以按下面的形式直接

调用

```
[Sc,Scl]=h2lqg(Stss)          % H2 控制器设计
[Sc,Scl]=hinf(Stss)           % H∞ 控制器设计
[γ,Sc,Scl]=hinfopt(Stss)      % H∞ 最优控制器设计
```

其中返回的变量 S_c 和 S_{cl} 分别为控制器模型和闭环系统状态方程模型的树结构表示，可以用 branch() 函数提取状态方程参数，再调用控制系统工具箱中的 ss() 函数转换成的状态方程模型。最优 $\mathcal{H}_\infty$ 控制器设计返回的 γ 是在加权函数下能获得的最小的 γ 值。

例 8-6 考虑例 8-3 中增广的系统模型，试分别设计出 $\mathcal{H}_2$ 控制器、$\mathcal{H}_\infty$ 控制器和 $\mathcal{H}_\infty$ 最优控制器，比较控制效果。

求解 和前面的例子一样，可以先构造出增广的状态方程模型，然后调用现成函数设计出控制器

```
>> A=[0,1,0,0; -5000,-100/3,500,100/3; 0,-1,0,1; 0,100/3,-4,-60];
   B=[0; 25/3; 0; -1]; C=[0,0,1,0]; D=0; G=ss(A,B,C,D); s=tf('s');
   W1=100/(s+1); W2=1e-5; W3=s/1000; G1=augtf(G,W1,W2,W3);
   Gc1=h2lqg(G1); Gc1=zpk(Gc1),  % 设计最优 H2 控制器
   Gc2=hinf(G1); Gc2=zpk(Gc2)    % 设计 H∞ 控制器
   [g,Gc3]=hinfopt(G1); g, Gc3=zpk(Gc3)  % 设计最优 H∞ 控制器
```

其实更一般地，由于 G_1 是状态方程变量，不是树变量，故设计出来的 G_c 自然就是状态方程模型，无需 branch() 函数提取，所以整个显示语句用 G_c=zpk(hinf(G_1)) 代替即可。由最优 $\mathcal{H}_\infty$ 控制器设计函数可以得出 $\gamma = 2.7031$。由上面的语句可以设计出各种控制器为

$$G_2(s) = \frac{-9945947.5203(s+67.4)(s+0.06391)(s^2+25.87s+4643)}{(s+1)(s^2+23.81s+535.7)(s^2+1370s+5.045\times10^5)}$$

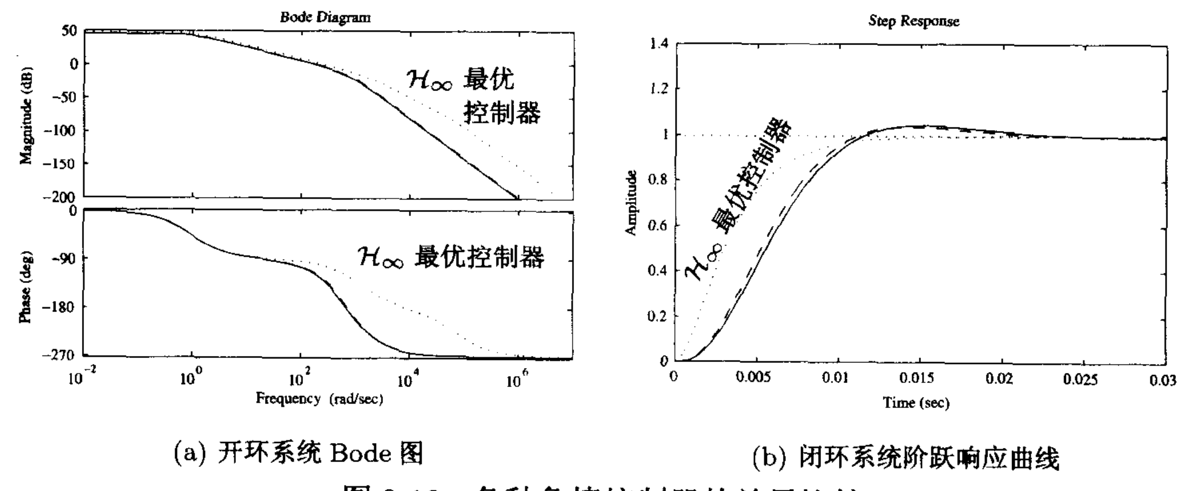
$$G_\infty(s) = \frac{-11633633.6883(s+67.4)(s+0.06391)(s^2+25.87s+4643)}{(s+1)(s^2+23.81s+535.7)(s^2+1419s+5.658\times10^5)}$$

$$G_{\rm opt}(s) = \frac{-2592334405.8282(s+67.4)(s+0.06391)(s^2+25.87s+4643)}{(s+7.007\times10^4)(s+1282)(s+1)(s^2+23.79s+535.7)}$$

其实，$\mathcal{H}_\infty$ 控制器设计过程中还将显示大量附加信息，例如 hinf() 函数将分别检验 $\mathcal{H}_\infty$ 控制器存在的 4 个条件，并检验闭环系统的稳定性，而最优 $\mathcal{H}_\infty$ 控制器将显示搜索最优值过程的中间数据。也可以在调用这些函数时给出关闭显示的命令。

在控制器作用下系统的开环 Bode 图和闭环阶跃响应曲线分别如图 8-10 (a)、(b) 所示，对本例来说，$\mathcal{H}_2$ 控制器和 $\mathcal{H}_\infty$ 控制器的效果类似，而 $\mathcal{H}_\infty$ 最优控制器控制效果则有些改善。

```
>> bode(G*Gc1,'-',G*Gc2,'--',G*Gc3,':'), figure;
   step(feedback(G*Gc1,1),'-',feedback(G*Gc2,1),...
       '--',feedback(G*Gc3,1),':')
```



(a) 开环系统 Bode 图　　(b) 闭环系统阶跃响应曲线

图 8-10 各种鲁棒控制器的效果比较

例 8-7 考虑多变量受控对象模型

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{0.806s+0.264}{s^2+1.15s+0.202} & \dfrac{-15s-1.42}{s^3+12.8s^2+13.6s+2.36}\\ \dfrac{1.95s^2+2.12s+0.49}{s^3+9.15s^2+9.39s+1.62} & \dfrac{7.15s^2+25.8s+9.35}{s^4+20.8s^3+116.4s^2+111.6s+18.8}\end{bmatrix}$$

该系统模型可以由下面的语句直接输入。现在考虑混合灵敏度问题，即引入加权矩阵

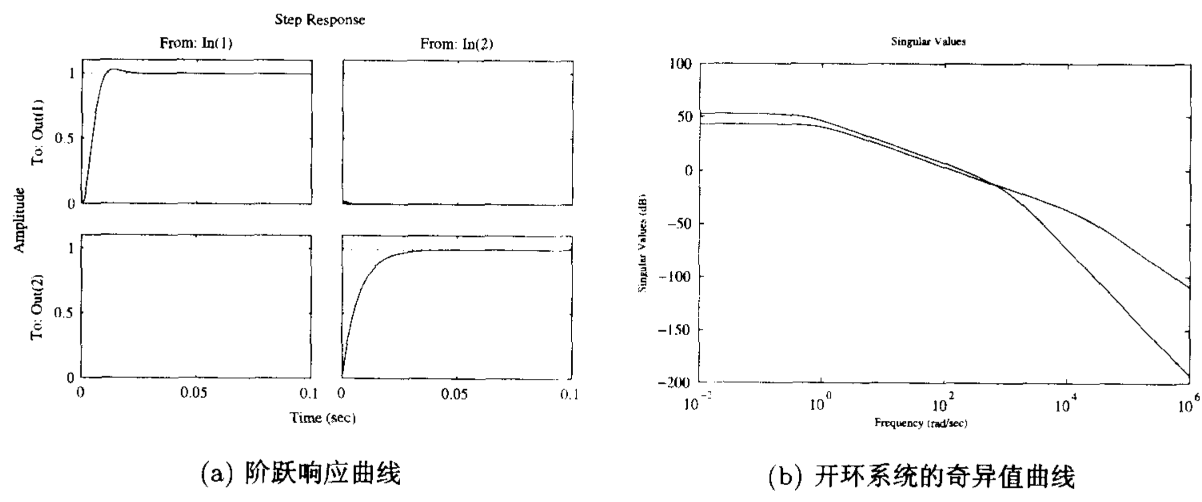
$$\boldsymbol{W}_1(s)=\begin{bmatrix}\dfrac{100}{s+0.5} & 0\\ 0 & \dfrac{100}{s+1}\end{bmatrix},\quad \boldsymbol{W}_3(s)=\begin{bmatrix}\dfrac{s}{1000} & 0\\ 0 & \dfrac{s}{200}\end{bmatrix} \tag{8-2-15}$$

试设计出最优 $\mathcal{H}_\infty$ 控制器，并探讨如何修正加权函数，改善控制效果。

求解 由于没有给出 $\boldsymbol{W}_2(s)$，所以和前面一样，可以设置 $\boldsymbol{W}_2(s)=\mathrm{diag}([10^{-5},10^{-5}])$。这样受控对象模型和增广的双端子模型用如下的语句就可以输入，并直接设计最优 $\mathcal{H}_\infty$ 控制器，绘制出该控制器作用下的阶跃响应曲线和开环系统的奇异值曲线，如图 8-11 所示。

```
>> s=tf('s'); g11=tf([0.806 0.264],[1 1.15 0.202]);
   g12=tf([-15 -1.42],[1 12.8 13.6 2.36]);
   g21=tf([1.95 2.12 0.49],[1 9.15 9.39 1.62]);
   g22=tf([7.15 25.8 9.35],[1 20.8 116.4 111.6 18.8]);
   G=[g11, g12; g21, g22]; W1=[100/(s+0.5),0; 0,100/(s+1)];
```

```
W2=[tf(1e-5),0; 0,tf(1e-5)]; W3=[s/1000,0; 0,s/200];
Tss=augtf(G,W1,W2,W3); [g,Gc]=hinfopt(Tss); zpk(Gc(1,2));
step(feedback(G*Gc,eye(2)),0.1), figure; sigma(G*Gc)
```



(a) 阶跃响应曲线　　(b) 开环系统的奇异值曲线

图 8-11　最优 $\mathcal{H}_\infty$ 控制器下的控制效果

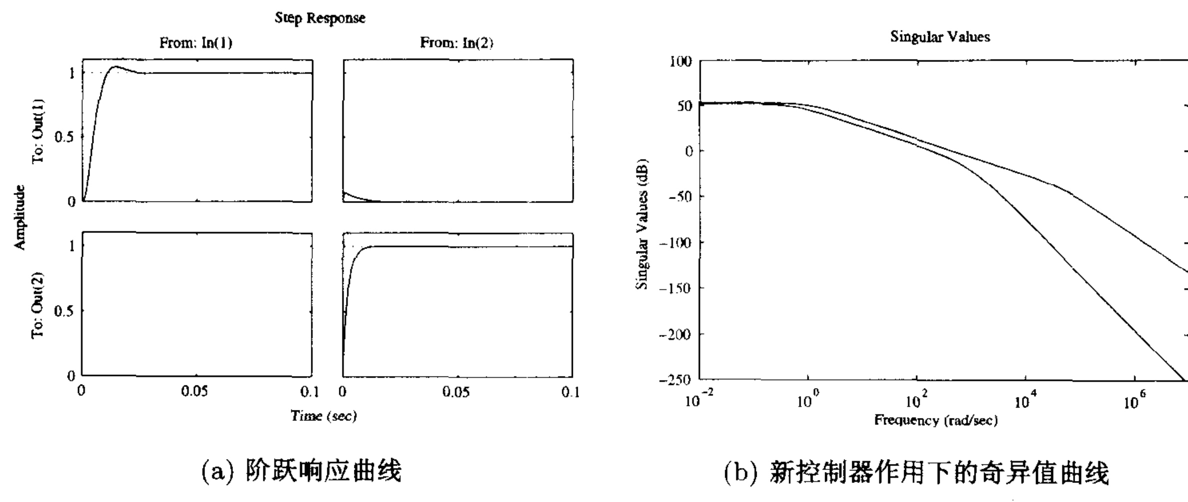
从得出的控制结果看，这样控制解决了第 4 章中未能很好解决的多变量系统的控制问题，得出的阶跃响应曲线是相当理想的，第 1 路阶跃输入作用于系统时能得出很好的 $y_1(t)$ 输入，而 $y_2(t)$ 几乎为 0，第 2 路输入单独作用时效果也相似。然而，这样设计出的控制器阶次是相当高的，其中 $g_{12}(s)$ 可以由上面的语句求出，其零极点表达式为以下的 14 阶模型

$$g_{12}(s)=\frac{\begin{array}{c}11185484.1011(s+1390)(s+572.6)(s+11.55)(s+8.002)\\(s+7.923)(s+0.9354)(s+0.9336)(s+0.9306)(s+0.5)\\(s+0.2175)(s+0.2164)(s+0.2147)(s+0.09967)\end{array}}{\begin{array}{c}(s+4.934\times10^4)(s+1713)(s+531.4)(s+11.55)(s+8.1)\\(s+1.052)(s+1)(s+0.9331)(s+0.9218)(s+0.5)(s+0.3369)\\(s+0.2467)(s+0.2263)(s+0.2167)\end{array}}$$

由得出的设计结果还可以看出，$y_{22}(t)$ 的响应速度和 $y_{11}(t)$ 相比较显得很慢，故需要加重 $\boldsymbol{W}_2(s)$ 的 $w_{1,22}(s)$ 权值，令 $w_{1,22}(s)=1000/(s+1)$，则可以重新设计最优 $\mathcal{H}_\infty$ 控制器，得出闭环系统的阶跃响应和开环奇异值曲线如图 8-12 所示。可见在新控制器下，$y_{22}(t)$ 效果明显改善。

```
>> W1=[100/(s+0.5),0; 0,1000/(s+1)]; Tss=augtf(G,W1,W2,W3);
   [g,Gc1]=hinfopt(Tss); step(feedback(G*Gc1,eye(2)),0.1);
   figure; sigma(G*Gc1)
```

由于控制器的阶次很高，在实际应用中难以实现，故可以考虑采用降阶算法降低控制器的阶次。从控制的效果看，即使用前面介绍的最优降阶算法对控制器



(a) 阶跃响应曲线　　(b) 新控制器作用下的奇异值曲线

图 8-12　修改 $\boldsymbol{W}_1(s)$ 后的控制效果

的各个子传递函数分别进行降阶，得出的效果也不会很理想，因为这样的模型降阶未考虑受控对象模型及闭环结构，所以应该采用闭环系统的控制器模型降阶的概念[8,9]，降低控制器的阶次，使其能直接实现。

8.2.5　增广系统模型与系统矩阵描述

在关于 $\mathcal{H}_\infty$ 控制器设计中，增广的系统模型是由双端口的状态方程形式描述的，而其他的设计方法中，包括 μ 分析与综合及线性矩阵不等式设计方法中，需要以系统矩阵的形式来描述。一般状态方程的矩阵描述形式为

$$\boldsymbol{P}=\left[\begin{array}{cc:c} & & n \\ \boldsymbol{A} & \boldsymbol{B} & \vdots \\ \boldsymbol{C} & \boldsymbol{D} & \\ & & 0 \\ \hdashline \multicolumn{2}{c:}{\boldsymbol{0}} & -\infty \end{array}\right] \tag{8-2-16}$$

若已知系统矩阵 $\boldsymbol{P}$，则可以由 `[`$\boldsymbol{A}$`,`$\boldsymbol{B}$`,`$\boldsymbol{C}$`,`$\boldsymbol{D}$`]=unpck(`$\boldsymbol{P}$`)` 直接提取出系统的状态方程模型，再由 `ss()` 函数就可以还原成状态方程对象模型。若已知系统的状态方程模型，直接采用该工具箱中提供的相应函数转换出系统矩阵是很烦琐的，所以可以考虑用下面的语句直接进行转换。

```
function P=ss2smat(G)
n=length(G.a); [p,q]=size(G.d)
P=[G.a G.b; G.c G.d]; P(1,n+q+1)=n; P(n+p+1,n+q+1)=-Inf;
```

例 8-8 试将例 8-3 中给出的对象模型和加权函数表示形式转换成系统矩阵形式。

求解 用下面的语句可以得出其系统矩阵模型 $\boldsymbol{P}$

```
>> A=[0,1,0,0; -5000,-100/3,500,100/3; 0,-1,0,1; 0,100/3,-4,-60];
```

```
B=[0; 25/3; 0; -1];  C=[0,0,1,0]; D=0; G=ss(A,B,C,D);
W1=[0,100; 1,1]; W2=1e-5; W3=[1,0; 0,1000];
S_tss=augtf(G,W1,W2,W3); P=ss2smat(S_tss)
```

经过这样的转换，可以立即得出系统矩阵的表示为

$$
\boldsymbol{P}=\left[\begin{array}{ccccc:cc:c}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 5 \\
-5000 & -33.333 & 500 & 33.333 & 0 & 0 & 8.3333 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 33.333 & -4 & -60 & 0 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 10^{-5} & 0 \\
0 & -0.001 & 0 & 0.001 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\infty
\end{array}\right]
$$

8.3 线性矩阵不等式理论与求解

线性矩阵不等式 (linear matrix inequalities，LMI) 的理论与应用是近 10 年来在控制界受到较广泛关注的领域[10]。线性矩阵不等式的概念及其在控制系统研究中的应用是由 Willems 提出的[11]，该方法的提出可以将很多控制中的问题变换成线性规划问题的求解，而线性规划问题的求解是很成熟的，所以由线性矩阵不等式的来求解控制问题是很有意义的。

本节将首先给出线性矩阵不等式的基本概念和常见形式，介绍必要的变换方法，然后介绍基于 MATLAB 中鲁棒控制工具箱的线性矩阵不等式求解方法，之后介绍控制问题如何用线性矩阵不等式表示以及线性矩阵不等式在控制系统设计中的应用。

8.3.1 线性矩阵不等式的一般描述

线性矩阵不等式的一般描述为

$$
\boldsymbol{F}(\boldsymbol{x})=\boldsymbol{F}_0+x_1\boldsymbol{F}_1+\cdots+x_m\boldsymbol{F}_m<0 \tag{8-3-1}
$$

式中，$\boldsymbol{x}=[x_1,\cdots,x_m]^{\mathrm{T}}$ 为多项式系数向量，又称为决策向量。$\boldsymbol{F}_i$ 为实对称矩阵或复 Hermit 矩阵。整个矩阵不等式小于零表示 $\boldsymbol{F}(\boldsymbol{x})$ 为负定矩阵，该不等式的解 $\boldsymbol{x}$ 是凸集，亦即

$$
\boldsymbol{F}[\alpha\boldsymbol{x}_1+(1-\alpha)\boldsymbol{x}_2]=\alpha\boldsymbol{F}(\boldsymbol{x}_1)+(1-\alpha)\boldsymbol{F}(\boldsymbol{x}_2)<0 \tag{8-3-2}
$$

其中 $\alpha > 0, 1-\alpha > 0$。该解又称为可行解。这样的线性矩阵不等式可以作为最优化问题的约束调节。假设有两个线性矩阵不等式 $\boldsymbol{F}_1(\boldsymbol{x}) < 0$ 和 $\boldsymbol{F}_2(\boldsymbol{x}) < 0$，则可以如下构造出一个线性矩阵不等式

$$\begin{bmatrix} \boldsymbol{F}_1(\boldsymbol{x}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}_2(\boldsymbol{x}) \end{bmatrix} < 0 \tag{8-3-3}$$

这样多个线性矩阵不等式可以写成一个单一的线性矩阵不等式。类似地，多个线性矩阵不等式 $\boldsymbol{F}_i(\boldsymbol{x}) < 0, i = 1, 2, \cdots, k$ 也可以合并成一个单一的线性矩阵不等式 $\boldsymbol{F}(\boldsymbol{x}) < 0$，其中

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{F}_1(\boldsymbol{x}) & & & \\ & \boldsymbol{F}_2(\boldsymbol{x}) & & \\ & & \ddots & \\ & & & \boldsymbol{F}_k(\boldsymbol{x}) \end{bmatrix} < 0 \tag{8-3-4}$$

线性矩阵不等式问题通常可以分为三类问题：可行解问题、线性目标函数最优化问题与广义特征值最优化问题，下面分别讲述。

1. 可行解问题

所谓可行解问题 (feasible solution problem) 就是最优化问题中的约束条件求解问题，即单纯求解不等式

$$\boldsymbol{F}(\boldsymbol{x}) < 0 \tag{8-3-5}$$

得出满足该不等式一个解的问题。求解线性矩阵不等式可行解事实上是求解 $\boldsymbol{F}(\boldsymbol{x}) < t_{\min}\boldsymbol{I}$，其中 $t_{\min}$ 是能够用数值方法找到的最小值。如果找到的 $t_{\min} < 0$，则得出的解是原问题的可行解，否则会提示无法找到可行解。

为演示一般控制问题和线性矩阵不等式之间的关系，首先考虑 Lyapunov 稳定性判定问题。对线性系统来说，若对给定的正定矩阵 $\boldsymbol{Q}$，方程

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} = -\boldsymbol{Q} \tag{8-3-6}$$

存在正定的解 $\boldsymbol{X}$，则该系统是稳定的。上述问题很自然地可以表示成对下面的 Lyapunov 不等式的求解问题。

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A} < 0 \tag{8-3-7}$$

由于 $\boldsymbol{X}$ 是对称矩阵，所以用 $n(n+1)/2$ 个元素构成的向量 $\boldsymbol{x}$ 即可以描述该矩阵

$$x_i = X_{i,1}, i = 1, \cdots, n, x_{n+i} = X_{i,2}, i = 2, \cdots, n, \cdots \tag{8-3-8}$$

该规律可以写成

$$x_{(2n-j+2)(j-1)/2+i}=X_{i,j},\ j=1,2,\cdots,n, i=j,j+1,\cdots,n \tag{8-3-9}$$

则给出 $\boldsymbol{x}$ 的下标即可以求出 i,j 的值。根据这样的思路可以编写出如下的 MATLAB 函数，该函数可以将 Lyapunov 方程转换为线性矩阵不等式

```
function F=lyap2lmi(A0)
if prod(size(A0))==1,
    n=A0; for i=1:n, for j=1:n,
       i1=int2str(i);j1=int2str(j); eval(['syms a' i1 j1]),
       eval(['A(' i1 ',' j1 ')=a' i1 j1,';'])
    end, end
else, n=size(A0,1); A=A0; end
vec=0;   for i=1:n, vec(i+1)=vec(i)+n-i+1; end
for k=1:n*(n+1)/2,  X=zeros(n);
    i=find(vec>=k); i=i(1)-1; j=i+k-vec(i)-1;
    X(i,j)=1; X(j,i)=1; F(:,:,k)=A.'*X+X*A;
end
```

该函数允许两种调用格式。若已知 $\boldsymbol{A}$ 矩阵，由 **F=lyap2lmi(A)**，则返回的 $\boldsymbol{F}$ 是三维数组，其第 i 层，即 $\boldsymbol{F}(:,:,i)$ 为所需的 $\boldsymbol{F}_i$ 矩阵。若只想得出$n\times n$ 的 $\boldsymbol{A}$ 矩阵转换出的线性矩阵不等式，则 **F=lyap2lmi(n)**，这时得出的 $\boldsymbol{F}$ 仍为上述定义的三维数组。在程序中，若使 $x_i=1$，而其他的 x_i 的值都为 0，则可以求出 $\boldsymbol{F}_i$ 矩阵。

例 8-9 若 $\boldsymbol{A}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$，试求出其线性矩阵不等式表示。若 $\boldsymbol{A}$ 为一般 3×3 实矩阵，试得出相应的线性矩阵不等式。

求解 输入 $\boldsymbol{A}$ 矩阵，再给出求解语句

```
>> A=[1,2,3; 4,5,6; 7,8,0]; F=lyap2lmi(A)
```

则可以得出 $\boldsymbol{F}_i$ 矩阵分别为

$$\begin{bmatrix}2&2&3\\2&0&0\\3&0&0\end{bmatrix},\begin{bmatrix}8&6&6\\6&4&3\\6&3&0\end{bmatrix},\begin{bmatrix}14&8&1\\8&0&2\\1&2&6\end{bmatrix},\begin{bmatrix}0&4&0\\4&10&6\\0&6&0\end{bmatrix},\begin{bmatrix}0&7&4\\7&16&5\\4&5&12\end{bmatrix},\begin{bmatrix}0&0&7\\0&0&8\\7&8&0\end{bmatrix}$$

若研究一般 3×3 矩阵，则可以给出如下命令

```
>> F=lyap2lmi(3)
```

这时得出的线性矩阵不等式为

$$x_1\begin{bmatrix}2a_{11}&a_{12}&a_{13}\\a_{12}&0&0\\a_{13}&0&0\end{bmatrix}+x_2\begin{bmatrix}2a_{21}&a_{22}+a_{11}&a_{23}\\a_{22}+a_{11}&2a_{12}&a_{13}\\a_{23}&a_{13}&0\end{bmatrix}+x_3\begin{bmatrix}2a_{31}&a_{32}&a_{33}+a_{11}\\a_{32}&0&a_{12}\\a_{33}+a_{11}&a_{12}&2a_{13}\end{bmatrix}$$

$$+x_4\begin{bmatrix}0 & a_{21} & 0\\ a_{21} & 2a_{22} & a_{23}\\ 0 & a_{23} & 0\end{bmatrix}+x_5\begin{bmatrix}0 & a_{31} & a_{21}\\ a_{31} & 2a_{32} & a_{33}+a_{22}\\ a_{21} & a_{33}+a_{22} & 2a_{23}\end{bmatrix}+x_6\begin{bmatrix}0 & 0 & a_{31}\\ 0 & 0 & a_{32}\\ a_{31} & a_{32} & 2a_{33}\end{bmatrix}<0$$

某些非线性的不等式也可以通过变换转换成线性矩阵不等式。其中，分块矩阵不等式的 Schur 补性质[12]是进行这样变换的常用方法。该性质的内容是：若某个仿射函数矩阵 $\boldsymbol{F}(\boldsymbol{x})$ 可以分块表示成 $\boldsymbol{F}(\boldsymbol{x})=\left[\begin{array}{c:c}\boldsymbol{F}_{11}(\boldsymbol{x}) & \boldsymbol{F}_{12}(\boldsymbol{x})\\ \hdashline \boldsymbol{F}_{21}(\boldsymbol{x}) & \boldsymbol{F}_{22}(\boldsymbol{x})\end{array}\right]$，其中 $\boldsymbol{F}_{11}(\boldsymbol{x})$ 是方阵，则下面三个矩阵不等式是等价的：

$$\boldsymbol{F}(\boldsymbol{x})<0 \tag{8-3-10}$$

$$\boldsymbol{F}_{11}(\boldsymbol{x})<0,\ \ \boldsymbol{F}_{22}(\boldsymbol{x})-\boldsymbol{F}_{21}(\boldsymbol{x})\boldsymbol{F}_{11}^{-1}(\boldsymbol{x})\boldsymbol{F}_{12}(\boldsymbol{x})<0 \tag{8-3-11}$$

$$\boldsymbol{F}_{22}(\boldsymbol{x})<0,\ \ \boldsymbol{F}_{11}(\boldsymbol{x})-\boldsymbol{F}_{12}(\boldsymbol{x})\boldsymbol{F}_{22}^{-1}(\boldsymbol{x})\boldsymbol{F}_{21}(\boldsymbol{x})<0 \tag{8-3-12}$$

例如，对一般代数 Riccati 方程稍加变换，则可以得出 Riccati 不等式

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}+(\boldsymbol{X}\boldsymbol{B}-\boldsymbol{C})\boldsymbol{R}^{-1}(\boldsymbol{X}\boldsymbol{B}-\boldsymbol{C}^{\mathrm{T}})^{\mathrm{T}}<0 \tag{8-3-13}$$

显然，该不等式因为含有二次项，所以它本身不是线性矩阵不等式。由 Schur 补性质可以看出，原非线性不等式可以等价地变换成

$$\boldsymbol{X}>0,\ \ \left[\begin{array}{c:c}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A} & \boldsymbol{X}\boldsymbol{B}-\boldsymbol{C}^{\mathrm{T}}\\ \hdashline \boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}-\boldsymbol{C} & -\boldsymbol{R}\end{array}\right]<0 \tag{8-3-14}$$

由二次型控制的要求已知，$\boldsymbol{R}=\boldsymbol{R}^{\mathrm{T}}>0$。显然，该矩阵含有未知矩阵 $\boldsymbol{X}$ 的二次型项，是非线性问题，不能直接表示成线性矩阵不等式的形式。

2. 线性目标函数最优化问题

考虑下面的最优化问题

$$\min_{\boldsymbol{x}\ \text{s.t.}\ \boldsymbol{F}(\boldsymbol{x})<0} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \tag{8-3-15}$$

由于约束条件是由线性矩阵不等式表示的，也就是第 5 章中介绍的线性约束，而目标函数也可以表示成由决策变量 $\boldsymbol{x}$ 构造的线性表示，所以这样的问题就是普通的线性规划求解问题。

考虑某控制系统状态方程模型 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$，其 $\mathcal{H}_\infty$ 范数可以通过 MATLAB 控制系统工具箱的 `norm()` 函数直接求解，该算法中采用了基于二分法的数值方程求解算法来计算系统的 $\mathcal{H}_\infty$ 范数。采用线性矩阵不等式方法也可以求出该系统的 $\mathcal{H}_\infty$ 范数。该范数即下面问题的解 γ[13]

$$\min_{\gamma,\ \boldsymbol{P}\ \text{s.t.}\ \left\{\begin{array}{l}\begin{bmatrix}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A} & \boldsymbol{P}\boldsymbol{B} & \boldsymbol{C}^{\mathrm{T}}\\ \boldsymbol{B}^{\mathrm{T}}\boldsymbol{P} & -\gamma\boldsymbol{I} & \boldsymbol{D}^{\mathrm{T}}\\ \boldsymbol{C} & \boldsymbol{D} & -\gamma\boldsymbol{I}\end{bmatrix}<0\\ \boldsymbol{P}>0\end{array}\right.} \gamma \tag{8-3-16}$$

3. 广义特征值最优化问题

广义特征值问题是线性矩阵不等式理论的一类最一般的问题。回顾第 3 章介绍的广义特征值问题，$\boldsymbol{Ax}=\lambda\boldsymbol{Bx}$，由该式演化可以得到更一般的不等式 $\boldsymbol{A}(\boldsymbol{x})<\lambda\boldsymbol{B}(\boldsymbol{x})$，可将 λ 看作矩阵的广义特征值，从而归纳出下面的最优化问题

$$\min_{\lambda,\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}\boldsymbol{A}(\boldsymbol{x})<\lambda\boldsymbol{B}(\boldsymbol{x})\\ \boldsymbol{B}(\boldsymbol{x})>0\\ \boldsymbol{C}(\boldsymbol{x})<0\end{array}\right.}\lambda \tag{8-3-17}$$

另外还可以有其他约束，归类成 $\boldsymbol{C}(\boldsymbol{x})<0$。在这样约束条件求取最小的广义特征值的问题可以由一类特殊的线性矩阵不等式来表示。事实上，若将这几个约束归并成单一的线性矩阵不等式，则这样的最优化问题和线性目标函数最优化问题是同样的问题。

8.3.2 线性矩阵不等式问题的 MATLAB 求解

早期的 MATLAB 中提供了线性矩阵不等式工具箱，可以直接求解相应的问题。新版本的 MATLAB 中将该工具箱并入了鲁棒控制工具箱，调用该工具箱中的函数可以求解线性矩阵不等式的各种问题。

描述线性矩阵不等式的方法是较烦琐的，用鲁棒控制工具箱中相应的函数描述这样的问题也是比较烦琐的。这里将介绍相关 MATLAB 语句的调用方法，并将给出例子演示相关函数的使用方法。

描述线性矩阵不等式应该有几个步骤：

① **创建 LMI 模型**：若想描述一个含有若干的 LMI 的整体线性矩阵不等式问题，需要首先调用 `setlmis([])` 函数来建立 LMI 框架，这样将在 MATLAB 工作空间中建立一个 LMI 模型框架。

② **定义需要求解的变量**：未知矩阵变量可以由 `lmivar()` 函数来申明，该函数的调用格式为 $\boldsymbol{P}$=`lmivar(key,[`$\boldsymbol{n}_1$,$\boldsymbol{n}_2$`])`，其中 `key` 是未知矩阵类型的标记，若 `key` 的值为 2，则变量 $\boldsymbol{P}$ 表示为 $n_1\times n_2$ 的一般矩阵。若 `key` 为 1，则 $\boldsymbol{P}$ 矩阵为 $n_1\times n_1$ 的对称矩阵。若 `key` 为 1，且 $\boldsymbol{n}_1$ 和 $\boldsymbol{n}_2$ 为向量，则 $\boldsymbol{P}$ 为块对角对称矩阵。`key` 值取 3 则表示 $\boldsymbol{P}$ 为特殊类型的矩阵。

③ **描述分块形式给出线性矩阵不等式**：申明了需求解的变量名后，可以由 `lmiterm()` 函数来描述各个 LMI 式子，该函数的调用格式比较复杂

`lmiterm([`k,i,j,$\boldsymbol{P}$`],`$\boldsymbol{A}$,$\boldsymbol{B}$`,flag)`

其中 k 为 LMI 编号，一个线性矩阵不等式问题可以由若干个 LMI 构成，用这样的方法可以分别描述各个 LMI。k 取负值时表示不等号 $<$ 右侧的项。

一个 LMI 子项可以由多个 lmiterm() 函数来描述。若第 k 个 LMI 是以分块形式给出的，则 i,j 表示该分块所在的行和列号。$\boldsymbol{P}$ 为已经由 lmivar() 函数申明过的变量名。$\boldsymbol{A},\boldsymbol{B}$ 矩阵表示该项中变量 $\boldsymbol{P}$ 左乘和右乘的矩阵，即该项含有 $\boldsymbol{APB}$。$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 设置成 1 和 -1 则分别表示单位矩阵 $\boldsymbol{I}$ 或负单位阵 $-\boldsymbol{I}$。若 flag 选择为 's'，则该项表示对称项 $\boldsymbol{APB}+(\boldsymbol{APB})^{\mathrm{T}}$。如果该项为常数矩阵，则可以将相应的 $\boldsymbol{P}$ 设置为 0，同时略去 $\boldsymbol{B}$ 矩阵。

④ **完成 LMI 模型描述**：由 lmiterm() 函数定义了所有的 LMI 后，就可以用 getlmis() 函数来确定 LMI 问题的描述，该函数的调用格式为 G=getlmis 。

⑤ **求解 LMI 问题**：定义了 G 模型后，就可以根据问题的类型调用相应函数直接求解，具体的格式为

```
[t_min,x]=feasp(G,options,target)              % 可行解问题
[c_opt,x]=mincx(G,c,options,x_0,target)        % 线性目标函数问题
[λ,x]=gevp(G,nlfc,options,λ_0,x_0,target)      % 广义特征值问题
```

这样获得的解 $\boldsymbol{x}$ 是一个向量，可以调用 dec2mat() 函数将所需的解矩阵提取出来。控制选项 options 是由 5 个值构成的向量，其第一个量表示要求的求解精度，通常可以取为 10^{-5}。

例 8-10 考虑 Riccati 不等式 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{XA}+\boldsymbol{XBR}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{Q}<0$，其中

$$\boldsymbol{A}=\begin{bmatrix}-2&-2&-1\\-3&-1&-1\\1&0&-4\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}-1&0\\0&-1\\-1&-1\end{bmatrix},\ \boldsymbol{Q}=\begin{bmatrix}-2&1&-2\\1&-2&-4\\-2&-4&-2\end{bmatrix},\ \boldsymbol{R}=\boldsymbol{I}_2$$

试求出该不等式的一个正定可行解 $\boldsymbol{X}$。

求解 该不等式显然不是线性矩阵不等式，类似前面介绍的 Riccati 不等式，可以引用 Schur 补性质对其进行变换，得出分块的 LMI 表示为 $\left[\begin{array}{c:c}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{XA}+\boldsymbol{Q} & \boldsymbol{XB}\\ \hdashline \boldsymbol{B}^{\mathrm{T}}\boldsymbol{X} & -\boldsymbol{R}\end{array}\right]<0$。考虑到需要求出原不等式的正定解 $\boldsymbol{X}$，故除了上面变换后的 Riccati 不等式外还需要满足 $\boldsymbol{X}>0$。可以将 Riccati 不等式设置成不等式 1，正定不等式设置成不等式 2，这样使用 lmiterm() 函数时，只需将 k 设置成 1 和 2 即可。另外，根据 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数，可以假定 $\boldsymbol{X}$ 为 3×3 对称矩阵。这样就可以用下面几个语句建立并求解可行解问题。因为第 2 不等式为 $\boldsymbol{X}>0$，所以序号采用 -2。

```
>> A=[-2,-2,-1; -3,-1,-1; 1,0,-4]; B=[-1,0; 0,-1; -1,-1];
   Q=[-2,1,-2; 1,-2,-4; -2,-4,-2]; R=eye(2);
   setlmis([]);    % 建立空白的 LTI 框架
```

```
X=lmivar(1,[3 1]);  % 申明需要求解的矩阵 X 为 3×3 对称矩阵
lmiterm([1 1 1 X],A',1,'s') % (1,1) 分块，对称表示为 A^T X + XA
lmiterm([1 1 1 0],Q)        % (1,1) 分块后面补一个 Q 常数矩阵
lmiterm([1 1 2 X],1,B)      % (1,2) 分块，填写 XB
lmiterm([1 2 2 0],-1)       % (2,2) 分块，填写 -R
lmiterm([-2,1,1,X],1,1)     % 设置第 2 不等式，即不等式 X > 0
G=getlmis;                  % 完成 LTI 框架的设置
[tmin b]=feasp(G);          % 求解可行解问题
X=dec2mat(G,b,X)            % 提取解矩阵
```

这样可以得出 $t_{\min}=-0.2427$，原问题的可行解为

$$X=\begin{bmatrix}1.0329 & 0.4647 & -0.23583\\ 0.4647 & 0.77896 & -0.050684\\ -0.23583 & -0.050684 & 1.4336\end{bmatrix}$$

值得指出的是，可能是由于该工具箱本身的问题，如果在描述 LMI 时给出了对称项，如 lmiterm([1 2 1 $\boldsymbol{X}$],$\boldsymbol{B}$',1)，则该函数将得出错误的结果。所以在求解线性矩阵不等式问题时一定不能给出对称项。

例 8-11 考虑例 5-24 中给出的线性系统 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},D)$ 的 $\mathcal{H}_\infty$ 范数，其中

$$\boldsymbol{A}=\begin{bmatrix}-4 & -3 & 0 & -1\\ -3 & -7 & 0 & -3\\ 0 & 0 & -13 & -1\\ -1 & -3 & -1 & -10\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0\\ -4\\ 2\\ 5\end{bmatrix};\ \boldsymbol{C}=[0,\ 0,\ 4,\ 0],\ D=0$$

求解 输入该线性系统的状态方程模型，由 norm() 函数可以立即求出系统的 $\mathcal{H}_\infty$ 范数为 0.4639。该问题当然能用线性矩阵不等式方法求解，式 (8-3-16) 中给出了求解范数的线性矩阵不等式模型，这里有两个决策变量，γ 和 $\boldsymbol{P}$，有两个不等式，其中第一个不等式为 3×3 的分块矩阵不等式，这样由下面的语句可以得出所需的解为 0.4651。在求解语句中，c 向量是由 mat2dec() 函数指定的。

```
>> A=[-4,-3,0,-1; -3,-7,0,-3; 0,0,-13,-1; -1,-3,-1,-10];
   B=[0; -4; 2; 5]; C=[0,0,4,0]; D=0; G=ss(A,B,C,D); norm(G,inf)
   setlmis([]); P=lmivar(1,[4,1]); gam=lmivar(1,[1,1]);
   lmiterm([1 1 1 P],1,A,'s'), lmiterm([1 1 2 P],1,B),
   lmiterm([1 1 3 0],C'); lmiterm([1 2 2 gam],-1,1),
   lmiterm([1 2 3 0],D'); lmiterm([1 3 3 gam],-1,1);
   lmiterm([-2 1 1 P],1,1); H=getlmis; c=mat2dec(H,0,1);
   [a,b]=mincx(H,c); gam_opt=dec2mat(H,b,gam)
```

由于得出的结果和由 norm() 函数得出的稍有区别，所以很自然地引出问题：哪

个是准确的？严格说，哪个也不准确。用 norm() 函数中的二分法得出的是近似解，而用 mincx() 函数得出的解由于默认精度较低，所以应该将求解精度设置为 10^{-5}，这样可以得出更精确的范数值为 0.4640。

```
>> options=[1e-5,0,0,0,0];
   [a,b]=mincx(H,c,options); gam_opt=dec2mat(H,b,gam)
```

8.3.3 基于 YALMIP 工具箱的最优化求解方法

Johan Jöfberg 博士开发了一个基于符号运算工具箱编写的模型优化工具箱 YALMIP (yet another LMI package)[14]，该工具箱提供的线性矩阵不等式求解方法和鲁棒控制工具箱中的 LMI 函数相比要直观得多。该工具箱的演示程序中还介绍了其他相关的最优化问题求解方法❶。

YALMIP 工具箱提供了简单的决策变量表示方法，可以调用 sdpvar() 函数来表示，该函数的调用方法为

```
X=sdpvar(n)             % 对称方阵的表示方法
X=sdpvar(n,m)           % 长方型一般矩阵的表示方法
X=sdpvar(n,n,'full')    % 一般方阵的表示方法
```

这样定义的矩阵还可以进一步利用，例如，这样定义的向量还可以和 hankel() 函数联合使用，构造出 Hankel 矩阵。类似地，由 intvar() 和 binvar() 函数还可以定义整型变量和二进制变量，从而求解整数规划和 0-1 规划问题。

由该工具箱针对 sdpvar 型变量定义的 set() 函数还可以描述矩阵不等式。如果有若干个这样的矩阵不等式，可以用 + 号将联立的若干个不等式“加”起来。

当然使用类似的方法还可以定义目标函数，描述了矩阵不等式约束后就可以分别如下调用

```
s=solvesdp(F)              % 求解可行解问题
s=solvesdp(F,f)            % 求解一般最优化问题，其中 f 为目标函数
s=solvesdp(F,f,options)    % 允许设定选项，如算法选择
```

求解结束后，可以由 **X=double(X)** 语句提取得出的解矩阵。

例 8-12 利用 YALMIP 工具箱，例 8-10 中的问题可以由下面语句更简洁地求解相应的矩阵不等式问题

```
>> A=[-2,-2,-1; -3,-1,-1; 1,0,-4]; B=[-1,0; 0,-1; -1,-1];
   Q=[-2,1,-2; 1,-2,-4; -2,-4,-2]; R=eye(2); X=sdpvar(3);
   F=set([A'*X+X*A+Q, X*B; B'*X, -R]<0)+set(X>0);
```

❶ 免费工具箱，下载地址：http://control.ee.ethz.ch/~joloef/yalmip.php。

```
sol=solvesdp(F); X=double(X)
```

该函数得出的解和前面得出的完全一致。

例 8-13 用 YALMIP 工具箱重新求解例 8-11 中给出的系统的 $\mathcal{H}_\infty$ 范数。

求解 由下面的语句可以直接求出系统的 $\mathcal{H}_\infty$ 范数为 0.4640。

```
>> A=[-4,-3,0,-1; -3,-7,0,-3; 0,0,-13,-1; -1,-3,-1,-10];
   B=[0; -4; 2; 5]; C=[0,0,4,0]; D=0; gam=sdpvar(1); P=sdpvar(4);
   F=set([A*P+P*A',P*B,C'; B'*P,-gam,D'; C,D,-gam]<0)+set(P>0);
   sol=solvesdp(F,gam); double(gam)
```

例 8-14 试用 YALMIP 工具箱求解例 5-26 中给出的线性规划问题。

求解 为方便起见，将该问题重新表述如下

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} 2x_2+x_3+4x_4+2x_5\leqslant 54 \\ 3x_1+4x_2+5x_3-x_4-x_5\leqslant 62 \\ x_1,x_2\geqslant 0,\ x_3\geqslant 3.32,\ x_4\geqslant 0.678,\ x_5\geqslant 2.57 \end{cases}} (-2x_1-x_2-4x_3-3x_4-x_5)$$

显然，$\boldsymbol{x}$ 是一个 5×1 列向量，这样可以由下面语句求解原问题

```
>> x=sdpvar(5,1); A=[0 2 1 4 2; 3 4 5 -1 -1]; B=[54; 62];
   xm=[0,0,3.32,0.678,2.57]'; F=set(A*x<B)+set(x>xm);
   sol=solvesdp(F,-[2 1 4 3 1]*x); double(x)
```

由该函数可以立即得出问题的解 $\boldsymbol{x}=[19.785,0,3.32,11.385,2.57]^{\mathrm{T}}$，与前面得出的完全一致。如果将决策变量设置为整数，可以用 intvar() 定义，并用下面语句求解整数规划问题

```
>> x=intvar(5,1); F=set(A*x<B)+set(x>xm);
   options=sdpsettings('solver','bnb');    % 选择分枝定界法
   sol=solvesdp(F,-[2 1 4 3 1]*x,options); double(x)
```

解为 $\boldsymbol{x}=[19,0,4,10,5]^{\mathrm{T}}$，和例 5-33 中的结果一致。

8.3.4 多线性模型的同时镇定问题

假设线性系统由 $\dot{\boldsymbol{x}}=\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{B}_i\boldsymbol{u}$，$i=1,\cdots,m$，试求出状态反馈矩阵 $\boldsymbol{K}$，使得 $\boldsymbol{u}(t)=-\boldsymbol{K}\boldsymbol{x}(t)$，使得所有的闭环系统 $\boldsymbol{A}_i+\boldsymbol{B}_i\boldsymbol{K}$ 均稳定，这样的镇定问题称为同时镇定问题 (simultaneous stabilization problem)。

求解每个 Lyapunov 不等式

$$\boldsymbol{X}_i>0,\ (\boldsymbol{A}_i+\boldsymbol{B}_i\boldsymbol{K})^{\mathrm{T}}\boldsymbol{X}_i+\boldsymbol{X}_i(\boldsymbol{A}_i+\boldsymbol{B}_i\boldsymbol{K})<0 \tag{8-3-18}$$

都可以得出 $\boldsymbol{X}_i$ 使得该闭环系统稳定，但如何寻找一个统一的 $\boldsymbol{X}$，使得各个子系统都稳定呢？含有统一的 $\boldsymbol{X}$ 矩阵的 Lyapunov 不等式如下给出

$$\boldsymbol{X}>0,\ (\boldsymbol{A}_i+\boldsymbol{B}_i\boldsymbol{K})^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}(\boldsymbol{A}_i+\boldsymbol{B}_i\boldsymbol{K})<0 \tag{8-3-19}$$

在该不等式中，需要求解的变量为 $\boldsymbol{X}$ 和 $\boldsymbol{K}$ 矩阵，其余矩阵均为已知矩阵。由上面得出的不等式可见，因为其中含有 $\boldsymbol{X}$ 和 $\boldsymbol{K}$ 的乘积项。所以应该采用某种变换将其改写成线性矩阵不等式，然后可以对其求解，设计出能够同时镇定若干个受控对象的状态反馈控制器。

展开式 (8-3-19) 可见

$$\boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}_i+\boldsymbol{K}^{\mathrm{T}}\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{B}_i\boldsymbol{K}<0 \tag{8-3-20}$$

利用矩阵相合变换的性质，即 $\boldsymbol{PQP}^{\mathrm{T}}$ 不改变 $\boldsymbol{Q}$ 矩阵正定性的性质，对上述矩阵左乘 $\boldsymbol{X}^{-1}$，右乘 $(\boldsymbol{X}^{-1})^{\mathrm{T}}$，且 $(\boldsymbol{X}^{-1})^{\mathrm{T}}=\boldsymbol{X}^{-1}$，则上述矩阵不等式可以变换成

$$\boldsymbol{X}^{-1}\boldsymbol{A}_i^{\mathrm{T}}+\boldsymbol{A}_i\boldsymbol{X}^{-1}+\boldsymbol{X}^{-1}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{B}_i^{\mathrm{T}}+\boldsymbol{B}_i\boldsymbol{K}\boldsymbol{X}^{-1}<0 \tag{8-3-21}$$

记 $\boldsymbol{P}=\boldsymbol{X}^{-1}$，$\boldsymbol{Y}=\boldsymbol{K}\boldsymbol{X}^{-1}$，则矩阵不等式可以变换成如下的线性矩阵不等式

$$\boldsymbol{A}_i\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A}_i^{\mathrm{T}}+\boldsymbol{B}_i\boldsymbol{Y}+\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{B}_i^{\mathrm{T}}<0 \tag{8-3-22}$$

加上 $\boldsymbol{X}>0$，即 $\boldsymbol{P}^{-1}>0$ 这个线性矩阵不等式，整个问题总共可以转换成 $m-1$ 个 LMI 来描述，再对整个 LMI 问题求可行解，则可以得出 $\boldsymbol{P}$ 和 $\boldsymbol{Y}$，最终可以得出同时镇定的 $\boldsymbol{K}$ 矩阵。

例 8-15 假设已知两个系统模型为

$$\boldsymbol{A}_1=\begin{bmatrix}-1&2&-2\\-1&-2&1\\-1&-1&0\end{bmatrix},\ \boldsymbol{B}_1=\begin{bmatrix}-2\\1\\-1\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}0&2&2\\2&0&2\\2&0&1\end{bmatrix},\ \boldsymbol{B}_2=\begin{bmatrix}-1\\-2\\-1\end{bmatrix}$$

试得出两个子系统的同时镇定解，并设计出状态反馈矩阵 $\boldsymbol{K}$。

求解 该问题对应三个线性矩阵不等式，有两个变量 $\boldsymbol{P}$ 和 $\boldsymbol{Y}$，其中 $\boldsymbol{P}$ 为 3×3 对称矩阵，$\boldsymbol{Y}$ 为 1×3 行向量。这样，三个不等式可以分别写成

$$\begin{cases}\boldsymbol{P}^{-1}>0,\ \text{或等价地}\ \boldsymbol{P}>0\\ \boldsymbol{A}_1\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A}_1^{\mathrm{T}}+\boldsymbol{B}_1\boldsymbol{Y}+\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{B}_1^{\mathrm{T}}<0\\ \boldsymbol{A}_2\boldsymbol{P}+\boldsymbol{P}\boldsymbol{A}_2^{\mathrm{T}}+\boldsymbol{B}_2\boldsymbol{Y}+\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{B}_2^{\mathrm{T}}<0\end{cases}$$

这样，由下面的语句可以求解这三个联立线性矩阵不等式。

```
>> A1=[-1,2,-2; -1,-2,1; -1,-1,0]; B1=[-2; 1; -1];
   A2=[0,2,2; 2,0,2; 2,0,1]; B2=[-1; -2; -1];
   setlmis([]); P=lmivar(1,[3,1]); Y=lmivar(2,[1,3]);
   lmiterm([1,1,1,P],-1,1);
   lmiterm([2,1,1,P],A1,1,'s'), lmiterm([2,1,1,Y],B1,1,'s')
   lmiterm([3,1,1,P],A2,1,'s'), lmiterm([3,1,1,Y],B2,1,'s')
   G=getlmis; [a,b]=feasp(G); P=dec2mat(G,b,P),
   Y=dec2mat(G,b,Y), X=inv(P); K=Y*X
```

求解此问题可以得出如下的解

$$X=\begin{bmatrix}0.13987 & 0.024173 & 0.10595\\ 0.024173 & 0.084939 & -0.050311\\ 0.10595 & -0.050311 & 0.21682\end{bmatrix}$$

这时，可以得出状态反馈向量 $K=[2.0739, 0.5616, 2.4615]^{\rm T}$。

如果采用 YALMIP 工具箱，则用下面语句可以求解同时镇定问题。

```
>> P=sdpvar(3); Y=sdpvar(1,3);
   F=set(A1*P+P*A1'+B1*Y+Y'*B1'<0)+set(A2*P+P*A2'+B2*Y+Y'*B2'<0);
   F=F+set(P>0); sol=solvesdp(F);
   P=double(P); X=inv(P), Y=double(Y), K=Y*X
```

该解与前面的解完全一致，得出的状态反馈向量也完全一致。

8.3.5 基于 LMI 的鲁棒最优控制器设计

前面介绍的很多基于范数的鲁棒控制问题均可以表示成线性矩阵不等式问题，下面只列出其中几个典型问题，请详见文献 [13]。

① $\mathcal{H}_2$ **控制器设计** 对状态方程模型 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$，其 $\mathcal{H}_2$ 控制可以等效地表示为下面的线性矩阵不等式问题

$$\min_{\rho,\boldsymbol{X},\boldsymbol{W},\boldsymbol{Z}\ \text{s.t.}\left\{\begin{array}{l}\boldsymbol{A}\boldsymbol{X}+\boldsymbol{B}_2\boldsymbol{W}+(\boldsymbol{A}\boldsymbol{X}+\boldsymbol{B}_2\boldsymbol{W})^{\rm T}+\boldsymbol{B}_1\boldsymbol{B}_1^{\rm T}<0\\ \begin{bmatrix}\boldsymbol{Z} & \boldsymbol{C}\boldsymbol{X}+\boldsymbol{D}\boldsymbol{W}\\ (\boldsymbol{C}\boldsymbol{X}+\boldsymbol{D}\boldsymbol{W})^{\rm T} & -\boldsymbol{X}\end{bmatrix}<0\\ \text{trace}(\boldsymbol{Z})<\rho\end{array}\right.} \rho \tag{8-3-23}$$

② $\mathcal{H}_\infty$ **控制器设计** 基于状态反馈的 $\mathcal{H}_\infty$ 最优控制器可以转换成下面的线

性矩阵不等式形式

$$\min \rho \qquad (8\text{-}3\text{-}24)$$

$$\rho,\boldsymbol{X},\boldsymbol{W} \text{ s.t.} \begin{cases} \begin{bmatrix} \boldsymbol{AX}+\boldsymbol{B}_2\boldsymbol{W}+(\boldsymbol{AX}+\boldsymbol{B}_2\boldsymbol{W})^{\mathrm{T}} & \boldsymbol{B}_1 & (\boldsymbol{C}_1\boldsymbol{X}+\boldsymbol{D}_{12}\boldsymbol{W})^{\mathrm{T}} \\ \boldsymbol{B}_1^{\mathrm{T}} & -\boldsymbol{I} & \boldsymbol{D}_{11}^{\mathrm{T}} \\ \boldsymbol{C}_1\boldsymbol{X}+\boldsymbol{D}_{12}\boldsymbol{W} & \boldsymbol{D}_{11} & -\rho\boldsymbol{I} \end{bmatrix} < 0 \\ \boldsymbol{X} > 0 \end{cases}$$

这时的状态反馈矩阵$\boldsymbol{K}=\boldsymbol{W}\boldsymbol{X}^{-1}$。

基于输出反馈的 $\mathcal{H}_\infty$ 问题可以转换成如下的线性矩阵不等式的最优化问题[10]

$$\min \quad \gamma \qquad (8\text{-}3\text{-}25)$$

$$\gamma,\boldsymbol{S},\boldsymbol{R} \text{ s.t.} \begin{cases} \begin{bmatrix} \boldsymbol{N}_{12} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{AR}+\boldsymbol{RA}^{\mathrm{T}} & \boldsymbol{RC}_1^{\mathrm{T}} & \boldsymbol{B}_1 \\ \boldsymbol{C}_1\boldsymbol{R} & -\gamma\boldsymbol{I} & \boldsymbol{D}_{11} \\ \boldsymbol{B}_1^{\mathrm{T}} & \boldsymbol{D}_{11}^{\mathrm{T}} & -\gamma\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_{12} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} < 0 \\ \begin{bmatrix} \boldsymbol{N}_{21} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{AS}+\boldsymbol{SA}^{\mathrm{T}} & \boldsymbol{SB}_1 & \boldsymbol{C}_1^{\mathrm{T}} \\ \boldsymbol{B}_1^{\mathrm{T}}\boldsymbol{S} & -\gamma\boldsymbol{I} & \boldsymbol{D}_{11}^{\mathrm{T}} \\ \boldsymbol{C}_1 & \boldsymbol{D}_{11} & -\gamma\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_{21} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} < 0 \\ \begin{bmatrix} \boldsymbol{R} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{S} \end{bmatrix} \geqslant 0 \end{cases}$$

在 MATLAB 的鲁棒控制工具箱中专门提供了基于线性矩阵不等式的控制器设计函数。如果鲁棒控制问题的增广系统 (8-2-8) 可以由系统矩阵表示，则可以由鲁棒控制工具箱的 `hinflmi()` 函数直接设计该问题可以用 LMI 工具箱中的函数可以直接设计出最优 $\mathcal{H}_\infty$ 控制器，该函数的调用格式为

$[\gamma_{\mathrm{opt}},\boldsymbol{K}]$=`hinflmi`$(\boldsymbol{P},[p,q])$

其中 $\boldsymbol{P}$ 为系统矩阵，p, q 为输出和输入路数，设计出的 $\boldsymbol{K}$ 为控制器的系统矩阵，而 γ_{opt} 为最优的 γ 值。

例 8-16 这里仍采用例 8-3 中增广的系统模型，试用线性矩阵不等式方法设计鲁棒控制器，并与前面例子中的控制器相比较。

求解 首先用前面的方法得到加权的系统增广模型，然后用 ss2smat() 函数获得系统矩阵，再用 LMI 工具箱的现成函数即可直接设计出最优 $\mathcal{H}_\infty$ 控制器

```
>> A=[0,1,0,0; -5000,-100/3,500,100/3; 0,-1,0,1; 0,100/3,-4,-60];
   B=[0; 25/3; 0; -1]; C=[0,0,1,0]; D=0; G=ss(A,B,C,D);
   W1=[0,100; 1,1]; W2=1e-5; W3=[1,0; 0,1000]; G1=augtf(G,W1,W2,W3);
   [g1,Gc1]=hinfopt(G1); Gc1=zpk(Gc1),  % 设计最优 H∞ 控制器
   P=ss2smat(G1); [g,K]=hinflmi(P,[1,1]); [a,b,c,d]=unpck(K);
```

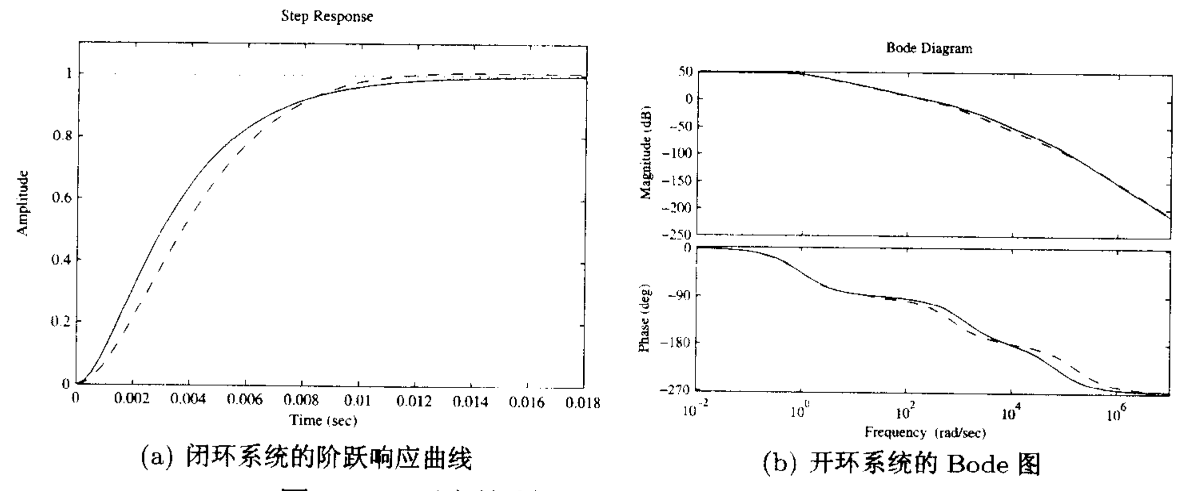
```
Gc2=zpk(ss(a,b,c,d))
step(feedback(G*Gc1,1),'-',feedback(G*Gc2,1),'--')
figure; bode(G*Gc1,'-',G*Gc2,'--')
```

得出的控制器为

$$G_{c1}(s)=\frac{-2592334405.8238(s+67.4)(s+0.06391)(s^2+25.87s+4643)}{(s+7.007\times10^4)(s+1282)(s+1)(s^2+23.79s+535.7)}$$

$$G_{c2}(s)=\frac{-3191219221.354(s+67.4)(s+0.06391)(s^2+25.87s+4643)}{(s+1.715\times10^5)(s+719.2)(s+0.9545)(s^2+22.34s+522.8)}$$

在这些控制器的作用下，闭环系统的阶跃响应曲线和开环系统的 Bode 图分别如图 8-13 (a)、(b) 所示。可见，在这两个控制器的作用下，控制效果是很接近的。



(a) 闭环系统的阶跃响应曲线　　(b) 开环系统的 Bode 图

图 8-13　两个控制器下系统的响应曲线比较

8.3.6　基于矩阵不等式约束的最优化求解方法

在控制系统问题求解中，线性矩阵不等式方法及工具有很重要的意义，但也有显著的局限性。因为它所能求解的问题都必须是能转换成 LMI 的问题，如果不是线性矩阵不等式则需要引入某种方法将其变换成相应的问题。从实际应用看，并非所有的问题都存在相应的转换方法，例如，在 Riccati 不等式中二次项的符号变成负号现在就没有现成的转换方法，而很多将矩阵的迹作为目标函数的应用也是不可取的，更准确的应该是矩阵的 1 范数等，而这样的范数不能表示成未知变量的线性函数，不能用线性矩阵不等式方法求解。如果采用矩阵的迹作为目标函数能表示成 $\boldsymbol{X}$ 的线性函数，但由于将对角元素纳入目标函数，势必会迫使非对角元素的值失去控制，变成很大的值，导致原始要求变形。

所以，对更一般的约束和更一般的目标函数，可以考虑第 5 章给出的非线性规划求解方法。诚然，非线性规划有时会无解或陷入局部最优解，在这样的情况

下可以考虑采用进化类优化算法，如第 7 章中介绍的遗传算法和粒子群优化算法，来求解问题。

8.4 定量反馈理论与设计方法

8.4.1 定量反馈理论概述

美国加州大学的 Issac Horowitz 教授和他的合作者们系统地提出了一种名为定量反馈理论 (quantitative feedback theory，简称 QFT) 的设计方法[15,16]。这些方法是基于频率响应的设计方法，可以用于带有很大对象不确定性的单变量系统、多变量系统以及各种高度非线性系统和时变系统的鲁棒设计，这种设计方法既可以用于最小相位系统的设计，也可以用于非最小相位系统的设计，QFT 方法提出得比较早，但过去并未引起很高的重视，近年来，在控制界重新又对 QFT 方法发生了兴趣，并出现了 MATLAB 的 QFT 设计工具箱[17]。虽然 QFT 方法可以用于多变量系统的设计，但该设计方法的核心是单变量最小相位系统的设计。本节将给出 QFT 的基本设计方法与工具箱概述。

8.4.2 单变量系统的 QFT 设计方法

由于单变量最小相位系统是 QFT 设计问题的核心，所以在这里将详细叙述单变量系统的 QFT 设计方法和思想。单变量系统的 QFT 设计步骤如下。

① **设定控制系统结构** QFT 设计下的二自由度控制系统结构如图 8-14 所示，在控制系统中 $\mathscr{P}(s)$ 为不确定的受控对象模型。$C(s)$ 为系统的控制器，QFT 控制的特点是该控制器是定常的，它可以控制含有大不确定性的对象，并可以对噪声干扰有满意的抑制作用。由于 $C(s)$ 的主要作用是保证系统能满足鲁棒稳定性的要求，其品质性能不一定很理想，这样常常需要引入前置滤波器 $F(s)$，其作用是动态地补偿系统的性能。

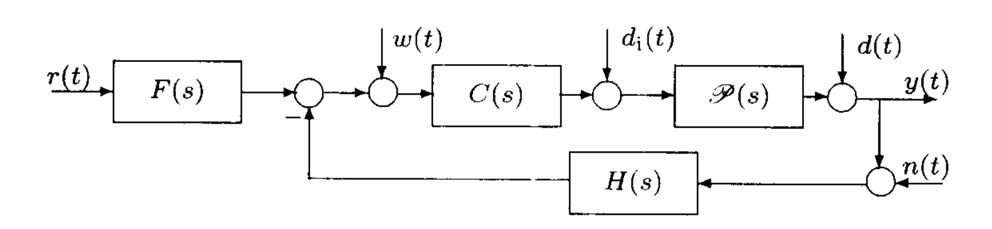
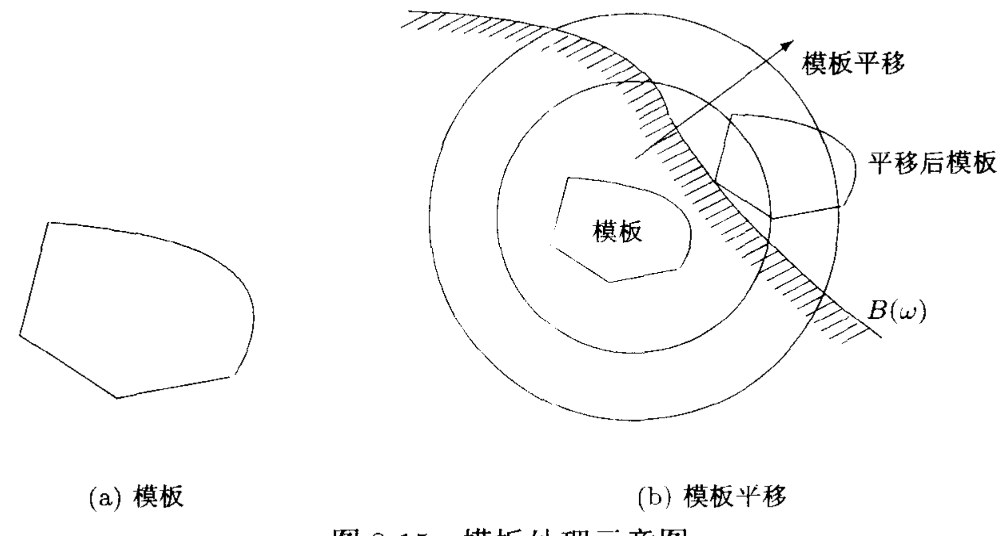


图 8-14 QFT 设计的控制系统结构图

② **构造频率响应模板** 选定一个频率点 ω_1，对此频率下的具有不确定性模型的各个选样模型进行频率响应分析，构造出 $\mathscr{P}(s)$ 的对象模板 (plant template)，如图 8-15 (a) 所示。



(a) 模板　　(b) 模板平移

图 8-15　模板处理示意图

申明了频率向量 $\boldsymbol{w}$，则可以由 plottmp() 函数绘制出各个频率下的模板。该函数的调用格式为 plottmp($\boldsymbol{w}$,$k_{\rm nom}$,$\mathscr{P}$) 。

在控制器 $C(s)$ 的作用下，系统的模板将发生平移现象，其中水平平移的幅度为 $C(s)$ 的相位角度 $\angle C(\mathrm{j}\omega_1)$，而垂直方向的平移量为控制器 $C(s)$ 的幅值 $|C(\mathrm{j}\omega_1)|$，如图 8-15 (b) 所示。

③ **给定设计要求** 在 QFT 设计中，为保证控制的效果，经常需要引入一些不等式来刻画研究的问题，如

- 鲁棒跟踪性能 在参数变化的情况下，最终闭环系统频率响应满足

$$T_{\rm m}(\omega)\leqslant\left|F(\mathrm{j}\omega)\frac{\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}{1+H(\mathrm{j}\omega)\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}\right|\leqslant T_{\rm M}(\omega) \tag{8-4-1}$$

- 鲁棒稳定性 系统模型满足

$$\left|\frac{\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}{1+H(\mathrm{j}\omega)\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}\right|\leqslant M_0 \tag{8-4-2}$$

除此之外，还可以有其他的不等式指标。所有可能的不等式指标在表 8-1 中列出。QFT 工具箱中提供了一个 sisobnds() 函数来描述各种不等式，该函数的调用格式为 bnd=sisobnds(type,$\boldsymbol{\omega}$,W,$\mathscr{P}$,C) ，其中 type 为表 8-1 中的不等式序号，$\boldsymbol{\omega}$ 为频率点向量，W 为表中相应的 W_i 对象，$\mathscr{P}$ 为不确定系统对象模型。

构造了多个 $\mathrm{bnd}_1,\cdots,\mathrm{bnd}_k$，则可以使用 grpbnds() 函数来构造整体的不等式边界变量 bnd=grpbnds($\mathrm{bnd}_1,\cdots,\mathrm{bnd}_k$) 。由整体的 bnd 变量

表 8-1 各种传递函数的边界约束

type	不等式约束	type	不等式约束
1	$\left\lvert\dfrac{\mathscr{P}CH}{1+\mathscr{P}CH}\right\rvert < W_1(s)$	2	$\left\lvert\dfrac{1}{1+\mathscr{P}CH}\right\rvert < W_2(s)$
3	$\left\lvert\dfrac{\mathscr{P}}{1+\mathscr{P}CH}\right\rvert < W_3(s)$	4	$\left\lvert\dfrac{G}{1+\mathscr{P}CH}\right\rvert < W_4(s)$
5	$\left\lvert\dfrac{CH}{1+\mathscr{P}CH}\right\rvert < W_5(s)$	6	$\left\lvert\dfrac{\mathscr{P}C}{1+\mathscr{P}CH}\right\rvert < W_6(s)$
7	$W_{7a}(s) < \left\lvert\dfrac{F\mathscr{P}C}{1+\mathscr{P}CH}\right\rvert < W_{7b}(s)$	8	$\left\lvert\dfrac{H}{1+\mathscr{P}CH}\right\rvert < W_8(s)$
9	$\left\lvert\dfrac{\mathscr{P}H}{1+\mathscr{P}CH}\right\rvert < W_9(s)$		

可以进一步用 `sectbnds()` 函数获得边界的交界：`ubnd=sectbnds(bnd)`。绘制各种边界的 MATLAB 函数为 `plotbnds(bnd)`。

④ **构造各个频率下的稳定下界** 对各个频率 $(\omega_2,\omega_3,\cdots,\omega_{\rm m})$ 也做相应的处理，则可以得出在各个频率下的下界曲线 $B(\omega_2),B(\omega_3),\cdots,B(\omega_{\rm m})$，这样若想得出最优的控制器，则可以由各个 $B(\omega_i)$ 曲线上选择一个点来构成 $L(\mathrm{j}\omega)$ 曲线，其中 $L(s)=\mathscr{P}(s)C(s)$。这样得出的控制器将极其复杂，所以要在 $B(\omega_i)$ 曲线允许的范围内得出“最优的”$L(s)$ 曲线，而不一定非得在边界线 $B(\omega_i)$ 上取点，由此可以设计出相应的 QFT 控制器 $C(s)$。该工作可以由用户界面 `lpshape()` 进行回路成型设计，该函数的调用格式比较特殊 `lpshape(`$\boldsymbol{\omega}$`,ubnd,`P_0`,`C_0`)`，其中 P_0 为标称受控对象模型，C_0 为控制器初始模型。该界面允许用户用交互的方式设计回路 $L(s)$ 的形状。确认 $L(s)$ 的形状后，单击 Apply 按钮来确认。再退出该界面。这时会出现一个对话框询问是否保存这样的回路，应该存成 `*.shp` 文件 (离散系统保存成 `*.dsh`)。

工具箱提供了 `getqft()` 函数从文件中读取回路成型设计的信息，生成 QFT 控制器 $C(s)$，具体的函数调用格式为 C`=getqft(文件名)`。

⑤ **设计 $F(s)$ 控制器** 按照前面的方法设计出来的控制器 $C(s)$ 往往不能满足式 (8-4-1) 中的频率要求，这样就需要一个前置滤波器 $F(s)$ 来使得系统满足该要求。

前置滤波器设计可以采用另外一个界面 `pfshape()` 来实现，该函数的调用格式为 `pfshape(type,`$\boldsymbol{\omega}$`,`W`,`$\mathscr{P}$`,`R`,`G`,`H`,`F_0`)`。设计完成后存储成 `*.fsh` 文件 (离散系统存成 `*.dfs`)。仍然可以使用 `getqft()` 函数提取前置滤波器模型。

下面用一个例子来演示 QFT 控制器设计的全过程。

例 8-17 考虑下面的不确定系统模型[17]

$$\mathscr{P}=\left\{\frac{k}{s(s+a)},\ k\in[1,2.5,6,10],a\in[1.5,3,6]\right\}$$

假设想设计出满足鲁棒边界要求

$$\left|\frac{\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}{1+H(\mathrm{j}\omega)\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}\right|\leqslant 1.2$$

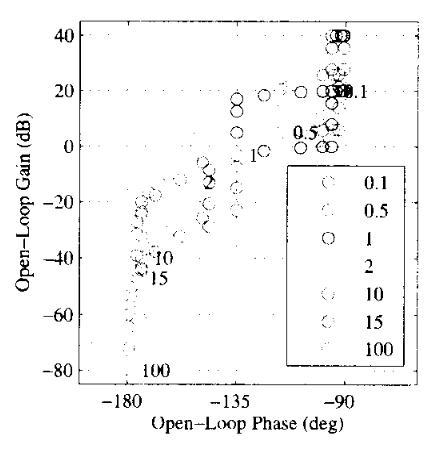
和低频 $(\omega<10)$ 时鲁棒性能要求

$$\frac{120}{s^3+17s^2+828s+120}\leqslant\left|F(\mathrm{j}\omega)\frac{\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}{1+H(\mathrm{j}\omega)\mathscr{P}(\mathrm{j}\omega)C(\mathrm{j}\omega)}\right|\leqslant\frac{0.6584(s+30)}{s^2+4s+19.752}$$

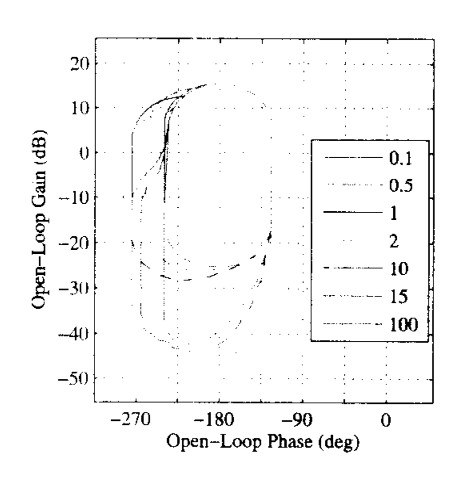
的 QFT 控制器，试调用 QFT 工具箱设计出这样的控制器。

求解 由下面的语句可以给出不确定系统模型，选择一组频率向量，则可以绘制出该不确定系统的模板图形，如图 8-16 (a) 所示。

```
>> c=1; a=1;
   for k=[1,2.5,6,10], P(1,1,c)=tf(k*a,[1,a,0]); c=c+1; end
   a=10; for k=[1,2.5,6,10], P(1,1,c)=tf(k*a,[1,a,0]); c=c+1; end
   k=1; for a=[1.5,3,6], P(1,1,c)=tf(k*a,[1,a,0]); c=c+1; end
   k=10; for a=[1.5,3,6], P(1,1,c)=tf(k*a,[1,a,0]); c=c+1; end
   w=[.1,.5,1,2,10,15,100]; k_nom=1; plottmpl(w,P,k_nom)
```



(a) 不同频率下的模板



(b) 频域响应边界

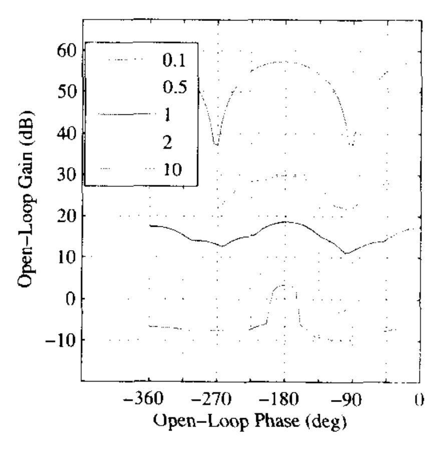
图 8-16 不确定系统的频域响应模板与边界

现在建立鲁棒边界模型并绘制其图形，如图 8-16 (b) 所示。

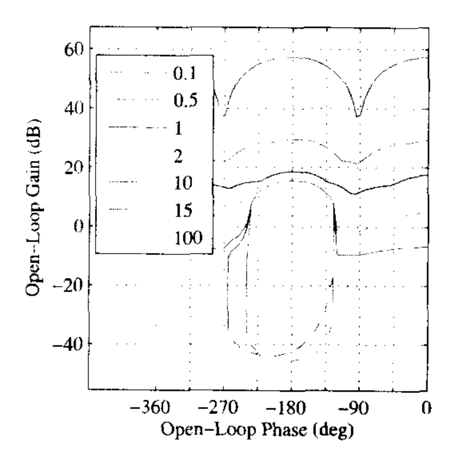
```
>> b1=sisobnds(1,w,1.2,P); plotbnds(b1)
```

下面的语句还可以绘制出鲁棒性能不等式的边界，如图 8-17 (a) 所示。注意，其中的 w(w<=10) 语句表示提取 $\omega \leqslant 10$ 的频率向量。由上述的两组边界可以提取出边界的交界，如图 8-17 (b) 所示。

```
>> Tm=tf(120,[1 17 82 120]); TM=tf(0.6584*[1 30],[1 4 19.752]);
   b2=sisobnds(7,w(w<=10),[TM;Tm],P); plotbnds(b2);
   figure; bnd=grpbnds(b1,b2); ubnd=sectbnds(bnd); plotbnds(ubnd);
```



(a) 鲁棒性能边界



(b) 边界交界

图 8-17 边界的进一步处理

假设已知初始控制器 $C_0(s)=\dfrac{100000(s+100)(s+1)}{(s+50)(s^2+1500s+10^6)}$，则可以由下面语句打开控制器设计界面，如图 8-18 所示。用户可以用鼠标拖动的方式修改 $B(\omega)$ 曲线，修改原则是不与交界边界冲突的前提下尽量偏向左侧，以得出尽可能快的响应速度。例如在这里初始控制器的前提下，将 Nichols 曲线尽量向上提，和交界相切。修正完该曲线后，可以单击 Apply 按钮，接受修改后的曲线。这时，可以关闭该曲线编辑界面，将设计存成相应的文件，如 c8qft1.shp。注意，这里的初始控制器参数形状选择非常重要，lpshape() 界面能提供的功能只是对曲线进行平移，改变其增益。

```
>> w1=logspace(-2,4,200); s=zpk('s');
   C0=100000*(s+100)*(s+1)/(s+50)/(s^2+1500*s+1e006);
   lpshape(w1,ubnd,P(1,1,1),C0);
```

由 getqft() 可以提取出控制器模型为 $C(s)=\dfrac{4708700.2369(s+100)(s+1)}{(s+50)(s^2+1500s+10^6)}$。

```
>> C=getqft('c8qft1.shp')
```

可以将前置补偿器 $F(s)$ 的初值选择为 $F_0(s)=\dfrac{10\omega_{\rm n}^2}{s^2+2\zeta\omega_{\rm n}s+\omega_{\rm n}^2}$ 的典型二阶系统模型，这样就可以用下面的语句来设计前置补偿器模型，得出如图 8-19 所示的界面。调节好了曲线后，就可以将其存成 c8qft1.fsh 文件。注意，如果选择 $\omega_{\rm n}=5$，则

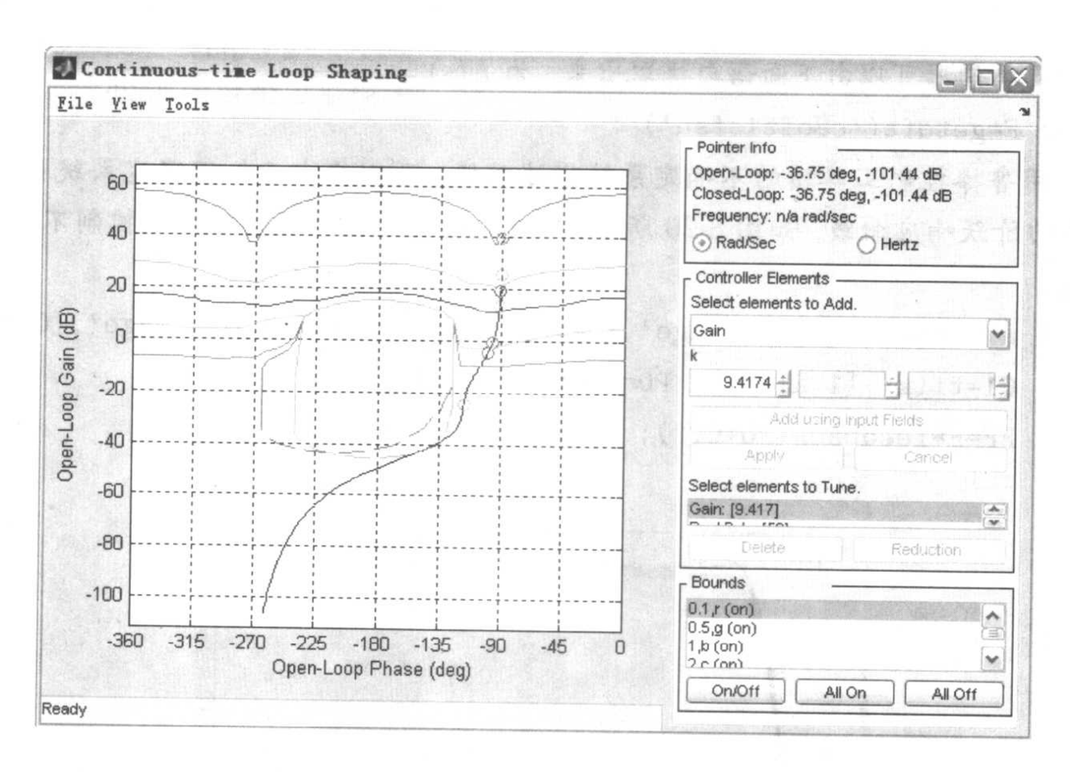


图 8-18 回路成型设计界面

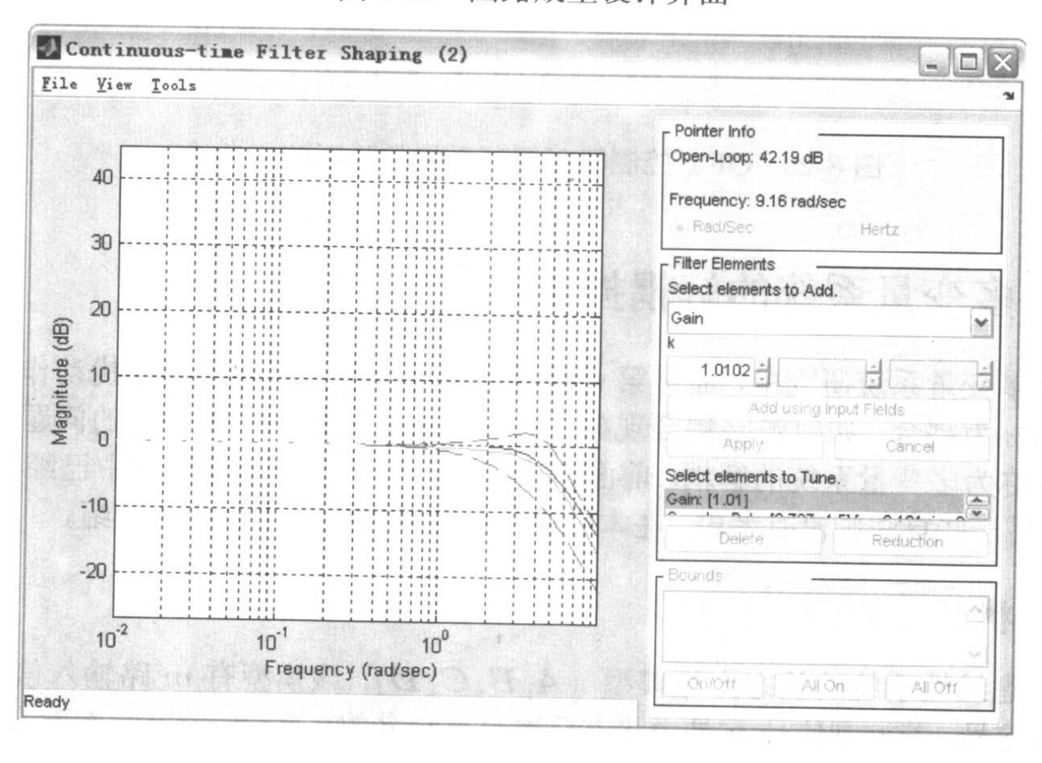


图 8-19 前置补偿器设计界面

无论如何调整，系统传递函数都将超出 $[T_{\rm m}(\omega), T_{\rm M}(\omega)]$ 边界，所以应该选择 $\omega_{\rm n}=4.5$。

```
>> wn=4.5; zet=0.707; F0=10*tf(wn^2,[1,2*zet*wn,wn^2]);
   pfshape(7,wl(w1<=10),[WM;Wm],P,[],C,[],F0);
```

前置补偿器可以由下面语句提取出来，为 $F(s) = \dfrac{20.1994}{s^2+6.363s+20.25}$。

```
>> F=getqft('c8qft1.fsh')
```

利用鲁棒控制工具箱的不确定系统描述功能，可以得出已知不确定系统在 QFT 控制下的阶跃响应曲线，如图 8-20 所示。可见，这样的控制器能很好地控制不确定系统模型。

```
>> a1=ureal('a',5,'range',[0,10]); k1=ureal('k',5,'range',[0,10]);
   G1=tf(k1,[1 a1 0]); P0=usample(G1,8);
   FF=F*feedback(P0*C,1); step(FF)
```

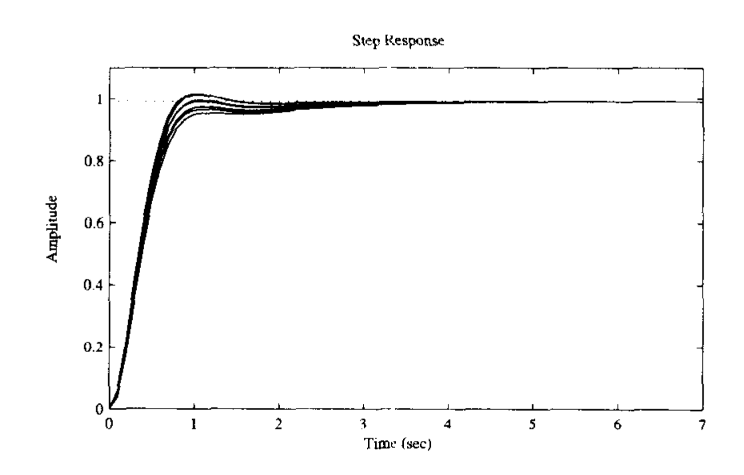


图 8-20 QFT 控制器下闭环系统的阶跃响应曲线

8.5 多变量系统的解耦控制

在多变量系统研究中，通常第 i 路控制输入对第 j 路输出存在扰动作用，这种现象称为耦合。如何消除耦合现象，是多年来系统解耦所需研究的问题。消除耦合又称为多变量系统的解耦。前面介绍的部分内容在控制器算法中已经考虑了解耦，另一些算法则没有考虑，在本节对一些解耦方法给出必要的介绍。

8.5.1 状态反馈解耦控制

考虑线性系统的状态方程模型 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D})$，该模型有 m 路输入信号，m 路输出信号。若控制信号 $\boldsymbol{u}$ 是由状态反馈建立起来的，即 $\boldsymbol{u}=\boldsymbol{\Gamma r}-\boldsymbol{Kx}$，这样闭环系统的传递函数矩阵模型可以写成

$$\boldsymbol{G}(s)=\Big[(\boldsymbol{C}-\boldsymbol{DK})(s\boldsymbol{I}-\boldsymbol{A}+\boldsymbol{BK})^{-1}\boldsymbol{B}+\boldsymbol{D}\Big]\boldsymbol{\Gamma} \tag{8-5-1}$$

对每个 $j, j=1,\cdots,m$ 定义出阶次 d_j，使得其为满足 $\boldsymbol{c}_j^{\mathrm{T}}\boldsymbol{A}^i\boldsymbol{B}\neq\boldsymbol{0},(i=0,1,\cdots,n-1)$ 的最小 j 值，则 $\boldsymbol{c}_j^{\mathrm{T}}$ 为矩阵 $\boldsymbol{C}$ 的第 j 行。

若 $m \times m$ 阶矩阵

$$\boldsymbol{B}_1 = \begin{bmatrix} \boldsymbol{c}_1^{\mathrm{T}} \boldsymbol{A}^{d_1} \boldsymbol{B} \\ \vdots \\ \boldsymbol{c}_m^{\mathrm{T}} \boldsymbol{A}^{d_m} \boldsymbol{B} \end{bmatrix} \tag{8-5-2}$$

为非奇异矩阵，若如下选择状态反馈矩阵 $\boldsymbol{K}$ 和前置矩阵 $\boldsymbol{\Gamma}$，则式 (8-5-1) 定义的系统可以动态解耦[18]。

$$\boldsymbol{\Gamma} = \boldsymbol{B}_1^{-1},\ \ \boldsymbol{K} = \boldsymbol{\Gamma} \begin{bmatrix} \boldsymbol{c}_1^{\mathrm{T}} \boldsymbol{A}^{d_1+1} \\ \vdots \\ \boldsymbol{c}_m^{\mathrm{T}} \boldsymbol{A}^{d_m+1} \end{bmatrix} \tag{8-5-3}$$

根据上述算法可以编写一个 MATLAB 函数 decouple() 来设计解耦矩阵。

```
function [G1,K,d,Gam]=decouple(G)
A=G.a; B=G.b; C=G.c; [n,m]=size(G.b); B1=[]; K0=[];
for j=1:m, for k=0:n-1
      if norm(C(j,:)*A^k*B)>eps, d(j)=k; break; end
   end
   B1=[B1; C(j,:)*A^d(j)*B]; K0=[K0; C(j,:)*A^(d(j)+1)];
end
Gam=inv(B1); K=Gam*K0; G1=tf(ss(A-B*K,B,C,G.d))*Gam;
```

该函数的调用格式为 $[\boldsymbol{G}_1,\boldsymbol{K},\boldsymbol{d},\boldsymbol{\Gamma}]$=decouple($\boldsymbol{G}$)，其中 $\boldsymbol{G}$ 为原始的多变量系统状态方程模型，$\boldsymbol{G}_1$ 为解耦后的传递函数矩阵，$\boldsymbol{K}$ 为状态反馈矩阵。向量 $\boldsymbol{d}$ 包含前面定义的 d_j 值，矩阵 $\boldsymbol{\Gamma}$ 为前置补偿器矩阵。

例 8-18 考虑下面的双输入双输出系统，试设计出满足完全解耦的状态反馈。

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 2.25 & -5 & -1.25 & -0.5 \\ 2.25 & -4.25 & -1.25 & -0.25 \\ 0.25 & -0.5 & -1.25 & -1 \\ 1.25 & -1.75 & -0.25 & -0.75 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 4 & 6 \\ 2 & 4 \\ 2 & 2 \\ 0 & 2 \end{bmatrix} \boldsymbol{u},\ \ \boldsymbol{y} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \end{bmatrix} \boldsymbol{x}$$

求解 系统的状态方程模型可以直接输入到系统中，这样就可以由下面命令立即设计出能够完全解耦的状态反馈矩阵 $\boldsymbol{K}$

```
>> A=[2.25, -5, -1.25, -0.5;  2.25, -4.25, -1.25, -0.25;
      0.25, -0.5, -1.25,-1;  1.25, -1.75, -0.25, -0.75];
   B=[4, 6; 2, 4; 2, 2; 0, 2]; C=[0, 0, 0, 1; 0, 2, 0, 2];
   D=zeros(2,2); G=ss(A,B,C,D); [G1,K,d,Gam]=decouple(G)
```

这样可以构造出状态反馈矩阵 $\boldsymbol{K}$ 和矩阵 $\boldsymbol{\Gamma}$。这时，传递函数矩阵 $\boldsymbol{G}_1(s)$ 可以实现完

全解耦。

$$G_1(s)=\begin{bmatrix}\frac{1}{s} & 0\\ 0 & \frac{1}{s}\end{bmatrix},\ K=\frac{1}{8}\begin{bmatrix}-1 & -3 & -3 & 5\\ 5 & -7 & -1 & -3\end{bmatrix},\ \Gamma=\begin{bmatrix}-1.5 & 0.25\\ 0.5 & 0\end{bmatrix}$$

引入状态反馈矩阵 K 与前置补偿器 Γ，则多变量系统可以完全解耦。解耦后的传递函数矩阵可以表示成 $G_1=\operatorname{diag}\left(\left[\frac{1}{s^{d_1+1}},\cdots,\frac{1}{s^{d_m+1}}\right]\right)$。

引入解耦补偿器 (K,Γ)，可以建立起如图 8-21 所示的反馈控制结构。因为虚线框中的部分实现了完全解耦，所以外环的控制器 $G_c(s)$ 可以分别有单独回路设计的方法实现。

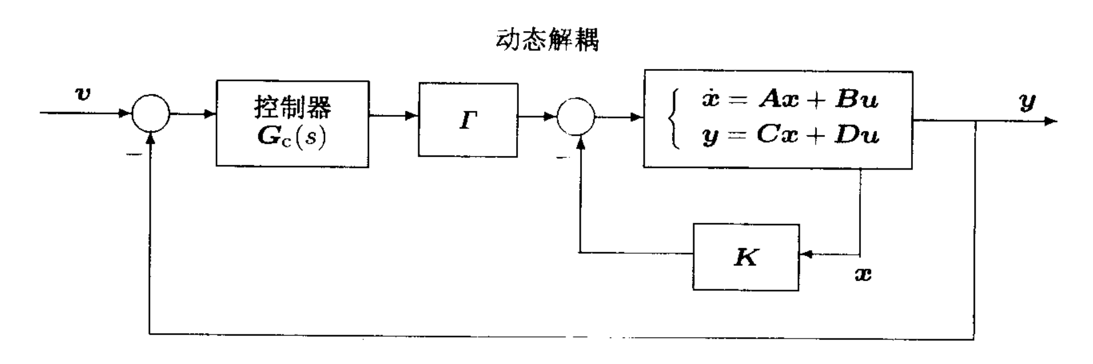


图 8-21 状态反馈控制器解耦结构

8.5.2 状态反馈的极点配置解耦系统

前面给出的动态解耦系统只能将多变量系统解耦成积分器型的对角传递函数矩阵，而积分器型受控对象在控制器设计中是很难解决的。如果仍想使用状态反馈型的解耦规则 $u=\Gamma r-Kx$，可以期望将解耦后的对角元素变成下面的形式

$$G_{K,\Gamma}(s)=\begin{bmatrix}\frac{1}{s^{d_1+1}+a_{1,1}s^{d_1}+\cdots+a_{1,d_1+1}} & & \\ & \ddots & \\ & & \frac{1}{s^{d_m+1}+a_{m,1}s^{d_m}+\cdots+a_{m,d_m+1}}\end{bmatrix} \tag{8-5-4}$$

其中 $d_i, i=1,\cdots,m$ 如前定义，每个多项式的系数 $s^{d_i+1}+a_{i,1}s^{d_i}+\cdots+a_{i,d_i+1}$ 可以用极点配置方法来设计。

可以考虑采用标准传递函数的形式来构造期望的多项式模型。满足 ITAE 最优准则的 n 阶标准传递函数由下式定义[19]

$$T(s)=\frac{1}{s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n} \tag{8-5-5}$$

表 8-2 ITAE 最优准则的标准传递函数分母多项式系数表

n	超调量	$\omega_n t_s$	分母多项式，其中 $a_{n+1}=\omega_n^n$
1			$s+\omega_n$
2	4.6%	6.0	$s^2+1.41\omega_n s+\omega_n^2$
3	2%	7.6	$s^3+1.75\omega_n s^2+2.15\omega_n^2 s+\omega_n^3$
4	1.9%	5.4	$s^4+2.1\omega_n s^3+2.4\omega_n^2 s^2+2.7\omega_n^3 s+\omega_n^4$
5	2.1%	6.6	$s^5+2.8\omega_n s^4+5.0\omega_n^2 s^3+5.5\omega_n^3 s^2+2.4\omega_n^4 s+\omega_n^5$
6	5%	7.8	$s^6+2.25\omega_n s^5+6.6\omega_n^2 s^4+8.6\omega_n^3 s^3+7.45\omega_n^4 s^2+2.95\omega_n^5 s+\omega_n^6$

其中，$T(s)$ 系统的分母多项式系数 a_i 在表 8-2 中给出。

根据前面的算法，可以容易地写出 n 阶标准传递函数模型的 MATLAB 函数 `std_tf()`

```
function G=std_tf(wn,n)
M=[1,1,0,0,0,0,0; 1,1.41,1,0,0,0,0;
   1,1.75,2.15,1,0,0,0; 1,2.1,3.4,2.7,1,0,0;
   1,2.8,5.0,5.5,3.4,1,0; 1,3.25,6.6,8.6,7.45,3.95,1];
G=tf(wn^n,M(n,1:n+1).*(wn*ones(1,n+1)).^[0:n]);
```

该函数的调用格式为 G=`std_tf`(ω_n,n)，其中 ω_n 为用户选定的自然频率，n 为预期的标准传递函数阶次。得出的 G 即标准传递函数模型。

定义一个矩阵 $\boldsymbol{E}$，使其每一行可以写成 $\boldsymbol{e}_i^{\mathrm{T}}=\boldsymbol{c}_i^{\mathrm{T}}\boldsymbol{A}^{d_i}\boldsymbol{B}$，另一个矩阵 $\boldsymbol{F}$ 的每一行 $\boldsymbol{f}_i^{\mathrm{T}}$ 可以定义为

$$\boldsymbol{f}_i^{\mathrm{T}}=\boldsymbol{c}_i^{\mathrm{T}}\left(\boldsymbol{A}^{d_i+1}+a_{i,1}\boldsymbol{A}^{d_i}+\cdots+a_{i,d_i+1}\boldsymbol{I}\right) \tag{8-5-6}$$

这样，状态反馈矩阵 $\boldsymbol{K}$ 和前置变换矩阵 $\boldsymbol{\Gamma}$ 可以写成

$$\boldsymbol{\Gamma}=\boldsymbol{E}^{-1},\quad \boldsymbol{K}=\boldsymbol{\Gamma}\boldsymbol{F} \tag{8-5-7}$$

基于本算法，可以写出极点配置动态解耦的 MATLAB 函数为

```
function [G1,K,d,Gam]=decouple_pp(G,wn)
A=G.a; B=G.b; C=G.c; [n,m]=size(G.b); E=[]; F=[];
for j=1:m, for k=0:n-1
        if norm(C(j,:)*A^k*B)>eps, d(j)=k; break; end
   end
   g1=std_tf(wn,d(j)+1); [n,cc]=tfdata(g1,'v');
   F=[F; C(j,:)*polyvalm(cc,A)]; E=[E; C(j,:)*A^d(j)*B];
end
Gam=inv(E); K=Gam*F; G1=tf(ss(A-B*K,B,C,G.d))*Gam;
```

该函数的调用格式为 $[\boldsymbol{G}_1,\boldsymbol{K},\boldsymbol{d},\boldsymbol{\Gamma}]$=`decouple_pp`$(\boldsymbol{G},\omega_n)$，其中，$\omega_n$ 为标准传递函数的自然频率，其他变量定义和前面给出的 `decouple()` 函数一致。

例 8-19 考虑例 8-18 中给出的多变量控制系统模型。选择 $\omega_\mathrm{n}=5$，试设计出极点配置的动态解耦器。

求解 可以由下面语句先输入系统状态方程模型，然后直接调用 decouple_pp() 函数来设计解耦器模型

```
>> A=[2.25, -5, -1.25, -0.5;  2.25, -4.25, -1.25, -0.25;
      0.25, -0.5, -1.25,-1;  1.25, -1.75, -0.25, -0.75];
   B=[4, 6; 2, 4; 2, 2; 0, 2]; C=[0, 0, 0, 1; 0, 2, 0, 2];
   D=zeros(2,2); G=ss(A,B,C,D); [G1,K,d,Gam]=decouple_pp(G,5)
```

这时，可以得出能够完全解耦的状态反馈控制器，其状态反馈矩阵 $\boldsymbol{K}$、前置补偿器 $\boldsymbol{\Gamma}$ 和解耦后的系统模型 $\boldsymbol{G}_1(s)$ 分别为

$$\boldsymbol{K}=\frac{1}{8}\begin{bmatrix}-1 & 17 & -3 & -35\\ 5 & -7 & -1 & 17\end{bmatrix},\ \boldsymbol{\Gamma}=\begin{bmatrix}-1.5 & 0.25\\ 0.5 & 0\end{bmatrix},\ \boldsymbol{G}_1(s)=\begin{bmatrix}\dfrac{1}{s+5} & \\ & \dfrac{1}{s+5}\end{bmatrix}$$

8.6 习题与思考题

1 试分别在 MATLAB 环境中描述下面的不确定模型，并绘制出频域响应曲线和时域响应曲线。

① $G(s)=\dfrac{s+1+\delta}{s^2-s+1}$，$\delta\in(-1.2,2)$

② $G(s)=\dfrac{c}{as^2+bs+c}$，其中 $a\in(0.1,5), b\in(0.2,2), c\in(1,10)$

③ $G(s)=\dfrac{(bs+1)\mathrm{e}^{-\tau s}}{s(s+a)}$，其中 $\tau\in(0.5,1], a\in(0.1,10)$，$b\in(-2,2)$。注意，由于不确定时间延迟模型不能直接用 usample() 函数表示，所以应该考虑采用其他近似方法，如 Padé 近似表示。

2 已知某时间延迟系统模型 $G(s)=\dfrac{\mathrm{e}^{-\tau s}}{s(s+a)}$，其中 $\tau\in[0,1]$，$a\in(1,5)$，试提取叠加型不确定性的频域响应包络线，并得出相应的传递函数模型。

3 试用最优化方法计算下面系统的 $\mathcal{H}_\infty$ 范数。

$$\boldsymbol{A}=\begin{bmatrix}-1 & 0 & 0 & 1\\ 0 & -1 & 4 & -3\\ 1 & -3 & -1 & -3\\ 0 & 4 & 2 & -1\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0 & 1\\ 0 & 0\\ -1 & 0\\ 0 & 0\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}-1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\end{bmatrix},\ \boldsymbol{D}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$$

4 假设对象模型为 $G(s)=\dfrac{1}{(0.01s+1)^2}$，并假定加权函数为 $W_1(s)=\dfrac{10}{s^3+2s^2+2s+1}$，

且 $W_3(s)=\dfrac{10s+1}{20(0.01s+1)}$，请完成下面的任务:

① 写出加权系统的双端子状态方程表示;

② 设计出一个最优的 $\mathcal{H}_\infty$ 控制器;

③ 绘制出在此控制器下闭环系统的阶跃响应与开环系统的 Nichols 图，并评价系统动态品质;

④ 设计一个最优 $\mathcal{H}_2$ 控制器，并比较控制效果。

5 已知对象模型 ① $G(s)=\dfrac{10}{(s+1)(s+2)(s+3)(s+4)}$，② $G(s)=\dfrac{10(-s+3)}{s(s+1)(s+2)}$。请设计出最小化灵敏度问题的最优的 $\mathcal{H}_\infty$ 控制器，在设计中可以使用标准函数的概念。对设计出来的系统进行时域与频域分析，并绘制出灵敏度函数、补灵敏度函数与加权灵敏度函数的幅频特性图。

6 在习题 5 的 ② 中，如果设计了最优 $\mathcal{H}_\infty$ 控制器，但对象模型的分母多项式变化为 $10(s+3)$，请在控制器不变的条件下分析系统的稳定性，并用时域和频域分析工具检验结果。

7 比较习题 5 中系统灵敏度问题的鲁棒控制器设计，请定性地分析标准传递函数的自然频率选择对系统响应的影响。

8 给习题 5 中的系统分别设计出灵敏度问题的最优 $\mathcal{H}_\infty$ 和 $\mathcal{H}_2$ 控制器，并对得出的系统进行时域与频域分析。

9 考虑文献 [20] 中给出的例子 $G(s)=\dfrac{-6.4750s^2+4.0302s+175.7700}{s(5s^3+3.5682s^2+139.5021s+0.0929)}$，若选择灵敏度加权函数 $W_1(s)=\dfrac{0.9}{1.0210}\dfrac{s^2+1.2s+1}{(s+0.001)(s+1.2)(0.001s+1)}$，试为系统设计一个 $\mathcal{H}_\infty$ 控制器，并仿真出闭环系统的阶跃响应曲线。

10 试写出习题 9 中模型的系统矩阵。

11 假设受控对象模型的标称模型 $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{0})$ 为

$$\boldsymbol{A}=\begin{bmatrix}-0.023 & -37 & -19 & -32\\ 0 & -1.9 & 0.98 & 0\\ 0.012 & -12 & -2.6 & 0\\ 0 & 0 & 1 & 0\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0 & 0\\ -0.41 & 0\\ -78 & 22\\ 0 & 0\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}0 & 57 & 0 & 0\\ 0 & 0 & 0 & 57\end{bmatrix}$$

实际受控对象为相乘型不确定系统 $\boldsymbol{G}(s)=\boldsymbol{G}(s)(\boldsymbol{I}+\boldsymbol{\Delta})$，根据不确定性和性能要求选择加权函数

$$\boldsymbol{W}_1(s)=\begin{bmatrix}\dfrac{0.5(s+\beta)}{s+0.03} & 0\\ 0 & \dfrac{0.5(s+\beta)}{s+0.03}\end{bmatrix},\ \boldsymbol{W}_3(s)=\begin{bmatrix}\dfrac{50(s+100)}{s+10000} & 0\\ 0 & \dfrac{50(s+100)}{s+10000}\end{bmatrix}$$

试设计出最优的 $\mathcal{H}_\infty$ 鲁棒控制器，并通过仿真选择合适的 β 值。

12 试分别采用鲁棒控制工具箱函数和 YALMIP 程序，按照线性矩阵不等式方式求解下面的线性规划问题。

① $$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} 4x_1-x_2+2x_3-x_4=-2 \\ x_1+x_2-x_3+2x_4\leqslant 14 \\ 2x_1-3x_2-x_3-x_4\geqslant -2 \\ x_{1,2,3}\geqslant -1, x_4 \text{ 无约束} \end{cases}} -3x_1+4x_2-2x_3+5x_4$$

② $$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1+x_2+x_3+x_4=4 \\ -2x_1+x_2-x_3-x_6+x_7=1 \\ 3x_2+x_3+x_5+x_7=9 \\ x_{1,2,\cdots,7}\geqslant 0 \end{cases}} x_6+x_7$$

13 试利用 LMI 工具箱求解下面的最优化问题。

$$\min_{\boldsymbol{X} \text{ s.t.} \begin{bmatrix} \boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}+\boldsymbol{Q} & \boldsymbol{X}\boldsymbol{B} \\ \boldsymbol{B}^{\mathrm{T}}\boldsymbol{X} & -\boldsymbol{I} \end{bmatrix} < 0} \operatorname{tr}(\boldsymbol{X})$$

其中 $\boldsymbol{A}=\begin{bmatrix} -1 & -2 & 1 \\ 3 & 2 & 1 \\ 1 & -2 & -1 \end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$，$\boldsymbol{Q}=\begin{bmatrix} 1 & -1 & 0 \\ -1 & -3 & -12 \\ 0 & -12 & -36 \end{bmatrix}$。

14 已知两个线性系统模型 $(\boldsymbol{A}_1,\boldsymbol{B}_1)$, $(\boldsymbol{A}_2,\boldsymbol{B}_2)$ 如下，试判定是否存在同一状态反馈向量，能够同时镇定这些系统模型。

$$\boldsymbol{A}_1=\begin{bmatrix} 4 & 2 & 0 & 1 \\ 3 & 3 & 2 & 3 \\ 1 & 2 & 1 & 0 \\ 4 & 4 & 4 & 1 \end{bmatrix}, \boldsymbol{B}_1=\begin{bmatrix} 2 \\ 2 \\ 2 \\ 3 \end{bmatrix}, \boldsymbol{A}_2=\begin{bmatrix} 0 & 4 & 3 & 1 \\ 4 & 3 & 2 & 2 \\ 3 & 1 & 0 & 2 \\ 3 & 1 & 0 & 4 \end{bmatrix}, \boldsymbol{B}_2=\begin{bmatrix} 4 \\ 4 \\ 2 \\ 4 \end{bmatrix}$$

15 利用线性矩阵不等式工具箱中相应的函数求解习题 9 中的鲁棒控制器设计问题，并和该习题的结果进行比较。

16 考虑习题 1 中给出的不确定模型，试采用定量反馈的方法，设计出合适的 QFT 控制器。

17 考虑不确定模型 $\mathscr{P}(s)=\dfrac{1}{s^2(s^2+2k)}$, $k\in(0.5,2)$，试设计出满足 $\left|\dfrac{\mathscr{P}C}{1+\mathscr{P}C}\right|<2.25$ 的控制器。

18 标准传递函数是在控制系统设计中很重要的一类传递函数模型，试得出不同阶次和不同自然频率下的标准传递函数模型及响应曲线。

19 考虑下面的双输入双输出系统模型[21]

① $\boldsymbol{A}=\begin{bmatrix} -1 & 1 & 1 & 1 \\ 6 & 0 & -3 & 1 \\ -1 & 1 & 1 & 2 \\ 2 & -2 & -2 & 0 \end{bmatrix}, \boldsymbol{B}=\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{C}=\begin{bmatrix} 2 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 \end{bmatrix}$

② $\boldsymbol{A}=\begin{bmatrix} 3 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}, \boldsymbol{B}=\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{C}=\begin{bmatrix} 2 & -1 & 1 \\ 0 & 2 & 1 \end{bmatrix}$

$$③\ \boldsymbol{G}(s)=\begin{bmatrix}\dfrac{3}{s^2+2} & \dfrac{2}{s^2+s+1}\\ \dfrac{4s+1}{s^2+2s+1} & \dfrac{1}{s}\end{bmatrix}$$

试求出能使其解耦的状态反馈方法，并考虑极点配置方式的解耦，讨论参考极点位置选择对解耦及控制的影响。

参考文献

[1] 梅生伟，申铁龙，刘康志. 现代鲁棒控制理论与应用 [M]. 北京: 清华大学出版社, 2003

[2] Zhou K, Doyle J. Essentials of robust control [M]. Prentice Hall, 1998

[3] Zames G. Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses [A]. Proceedings 17th Allerton Conference [C]. 1979, 744~752. Also, Transactions on Automatic Control, AC-26(2):585~601, 1981

[4] Skogestad S, Postlethwaite I. Multivariable feedback control: analysis and design [M]. New York: John Wiley & Sons, 1996

[5] De Cuyper J, Swevers J, Verhaegen M, et al. $\mathcal{H}_\infty$ feedback control for signal tracking on a 4 poster test rig in the automotive industry [A]. Proceedings of International Conference on Noise and Vibration Engineering [C]. 2000, 61~67

[6] Grimble M J. LQG optimal control design for uncertain systems [J]. Proceedings IEE, Part D, 1990, 139:21~30

[7] Doyle J C, Glover K, Khargonekar P, et al. State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems [J]. IEEE Transactions on Automatic Control, 1989, AC-34:831~847

[8] Anderson B D O. Controller design: moving from theory to practice [J]. IEEE Control Systems Magazine, 1993, 13(4):16~25. (Also Bode Prize Lecture, Conference on Decision and Control, Brighton, 1992)

[9] Anderson B D O, Liu Y. Controller reduction: concepts and approaches [J]. IEEE Transactions on Automatic Control, 1989, AC-34(8):802~812

[10] Boyd S, Ghaoui L El, Feron E, et al. Linear matrix inequalities in systems and control theory [M]. Philadelphia: SIAM books, 1994

[11] Willems J C. Least squares stationary optimal control and the algebraic Riccati equation [J]. IEEE Transactions on Automatic Control, 1971, 16(6):621~634

[12] Scherer C, Weiland S. Linear matrix inequalities in control [M]. Delft University of Technology, 2005

[13] 俞立. 鲁棒控制 — 线性矩阵不等式处理方法 [M]. 北京: 清华大学出版社, 2002

[14] Jöfberg J. YALMIP: a toolbox for modeling and optimization in MATLAB [A]. Proceedings of IEEE International Symposium on Computer Aided Control Systems Design [C]. Taipei, 2004, 284~289

[15] Horowitz I. Quantitative feedback theory (QFT) [J]. Proceedings IEE, Part D, 1982, 129:215~226

[16] Horowitz I. Survey of quantitative feedback theory (QFT) [J]. International Journal of Control, 1991, 53(2):255~291

[17] Borghesani C, Chait Y, Yaniv O. Quantitative feedback theory toolbox user's guide [Z]. Terasoft Inc, 2003

[18] Balasubramanian R. Continuous time controller design [M], IEE Control Engineering Series, volume 39. London: Peter Peregrinus Ltd, 1989

[19] Dorf R C, Bishop R H. Modern Control Systems (9th ed.) [M]. Upper Saddle River, NJ: Prentice-Hall, 2001

[20] Doyle J C, Francis B A, Tannerbaum A R. Feedback control theory [M]. New York: MacMillan Publishing Company, 1991

[21] 郑大钟. 线性系统理论 (第 2 版) [M]. 北京: 清华大学出版社, 2002

第 9 章 分数阶微积分学问题的计算机求解

一般地，$\mathrm{d}^n y/\mathrm{d}x^n$ 表示 y 对 x 的 n 阶导数，但若 $n = 1/2$ 时是什么含义呢？这是 300 多年以前法国著名数学家 Guillaume François Antoine L'Hôpital 问过微积分学创造者之一 Gattfried Wilhelm Leibnitz 的一个问题[1]。分数阶微积分理论建立至今已经有 300 年的历史了，但早期主要侧重于理论研究，近年来在很多领域都已经开始应用分数阶微积分学理论，如在自动控制领域出现的分数阶控制理论等新的分支。

以往的控制理论和其他数学建模方法侧重于集中参数系统的建模，例如电阻可以用一个比例系数来表示。在电阻不能用集中参数表示时，则需要用描述分布参数系统的偏微分方程来精确描述，例如远距离传输线的模型和电热炉模型等。这类模型在控制系统仿真软件中很难描述。引入分数阶微分算子，则可以将其用分数阶微分方程描述，在仿真回路中可以容易地表示这样的问题。另外，由于分数阶微积分本身的特性，分数阶控制器具有很多整数阶系统无法实现的优越性，所以研究分数阶系统的建模、分析与设计也是很有实际意义的。

本章 9.1 节将介绍分数阶微积分的定义及各种计算方法，还将给出分数阶微积分的各种性质。9.2 节将以分数阶线性系统建模为例，介绍 MATLAB 编程中一个很重要的面向对象设计的基本技术 —— 类的建立，将介绍分数阶传递函数类的设计。根据该传递函数类分别定义出系统模块各种连接方法的重载运算、时域分析和频域分析的概念 MATLAB 实现，为分数阶系统的分析打下基础。9.3 节将介绍利用著名的 Oustaloup 滤波器逼近分数阶微分算子的方法，并给出改进的微分算子滤波器，显著提高分数阶模型的拟合精度。采用这样的滤波器无疑会增加系统模型的阶次，会使得阶次高达数十或上百，使得系统的分析与设计变得困难，故 9.4 节将给出使得误差 $\mathcal{H}_2$ 范数最小化的模型次最优降阶算法。9.5 节将探讨分数阶系统的控制器设计方法。

9.1 分数阶微积分的定义

在分数阶微积分理论发展过程中，出现了很多种函数的分数阶微积分的定义，如由整数阶微积分直接扩展而来的 Cauchy 积分公式、Grünwald-Letnikov 分数阶微积分定义、Riemann-Liouville 分数阶微积分定义以及 Caputo 定义等，本节将先介绍这些定义及其等效关系，再给出分数阶微积分的各种性质。

9.1.1 分数阶微积分的定义

1. 分数阶 Cauchy 积分公式

分数阶 Cauchy 积分公式从简单整数阶积分直接扩展而来

$$\mathscr{D}^{\gamma} f(t)=\frac{\Gamma(\gamma+1)}{2\pi \mathrm{j}}\int_{\mathrm{C}}\frac{f(\tau)}{(\tau-t)^{\gamma+1}}\mathrm{d}\tau \tag{9-1-1}$$

其中，C 为包围 $f(t)$ 单值与解析开区域的光滑曲线。

由 Cauchy 积分公式可以容易地推导出正弦、余弦函数的整数阶导数公式

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[\sin at\right]=a^k\sin\left(at+\frac{k\pi}{2}\right),\ \ \frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[\cos at\right]=a^k\cos\left(at+\frac{k\pi}{2}\right) \tag{9-1-2}$$

可以证明，该公式同样适用于 k 为分数的情形。

2. Grünwald-Letnikov 分数阶微积分定义

Grünwald-Letnikov 分数阶微积分定义是分数阶控制中最广泛应用的分数阶微积分定义，该定义的数学表示为

$${}_a\mathscr{D}_t^{\alpha} f(t)=\lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{j=0}^{[(t-a)/h]}(-1)^j\binom{\alpha}{j}f(t-jh) \tag{9-1-3}$$

其中，$w_j^{(\alpha)}=(-1)^j\binom{\alpha}{j}$ 为函数 $(1-z)^{\alpha}$ 的多项式系数，该系数还可以更简单地由下面的递推公式直接求出：

$$w_0^{(\alpha)}=1,\ \ w_j^{(\alpha)}=\left(1-\frac{\alpha+1}{j}\right)w_{j-1}^{(\alpha)},\ j=1,2,\cdots \tag{9-1-4}$$

根据该定义可以推导出分数阶微分计算的算法为

$${}_a\mathscr{D}_t^{\alpha} f(t)=\lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{j=0}^{[(t-a)/h]}(-1)^j\binom{\alpha}{j}f(t-jh)\approx\frac{1}{h^{\alpha}}\sum_{j=0}^{[(t-a)/h]}w_j^{(\alpha)}f(t-jh) \tag{9-1-5}$$

假设步长 h 足够小，则可以用式 (9-1-5) 直接求出函数数值微分的近似值，并可以证明[2]，该公式的精度为 $o(h)$。所以，利用 Grünwald-Letnikov 定义可以立即编写出下面的函数来求取给定函数的分数阶微分函数[3]。

```
function dy=glfdiff(y,t,gam)
h=t(2)-t(1); dy(1)=0; y=y(:); t=t(:);
w=1; for j=2:length(t), w(j)=w(j-1)*(1-(gam+1)/(j-1)); end
for i=2:length(t), dy(i)=w(1:i)*[y(i:-1:1)]/h^gam; end
```

其中，$\boldsymbol{y}, \boldsymbol{t}$ 分别为给定函数的采样值与时刻值构成的向量。要求 $\boldsymbol{t}$ 为等间距向量，且 γ 为分数阶的阶次，这样得出的 $\boldsymbol{d}_{\rm y}$ 向量为函数的分数阶导数。

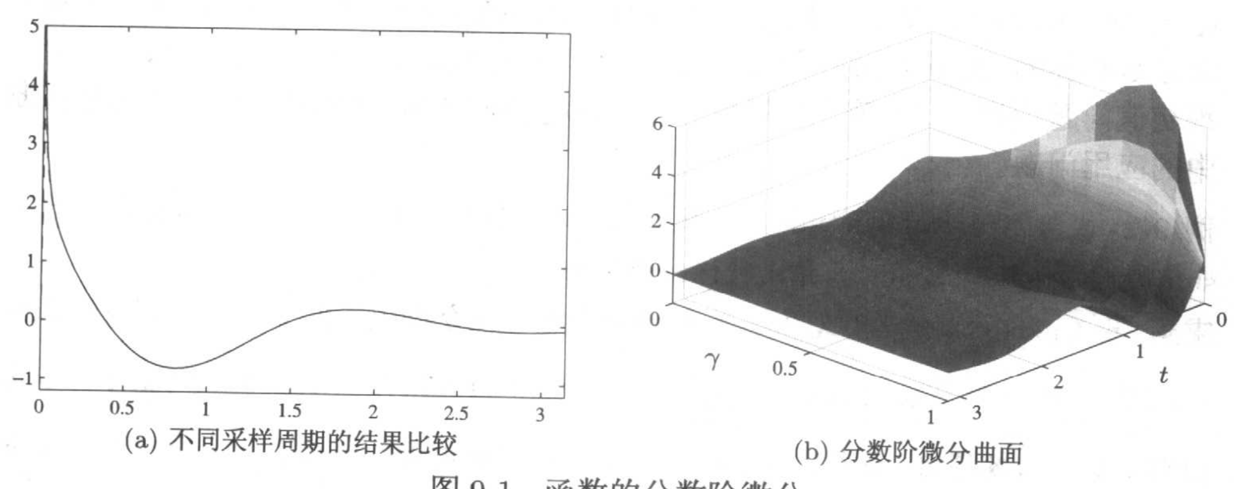
例 9-1 已知函数 $f(t)={\rm e}^{-t}\sin(3t+1)$, $t\in(0,\pi)$，试求出其分数阶导数。

求解 分别选择采样周期 $T=0.01$ 和 0.001，则可以得出在这两个采样周期下计算出的 0.5 阶导数函数曲线，如图 9-1 (a) 所示。可见，在 t 接近 0 的区域外二者是很接近的。对本函数而言，选择 $T=0.01$ 可以足够精确地求出函数的分数阶微分。

```
>> t1=0:0.001:pi; y1=exp(-t1).*sin(3*t1+1); dy1=glfdiff(y1,t1,0.5);
   t2=0:0.01:pi; y2=exp(-t2).*sin(3*t2+1); dy2=glfdiff(y2,t2,0.5);
   plot(t1,dy1,t2,dy2,'--'), axis([0,pi,-1.2,5])
```

对不同的 γ 选值，可以调用下面的语句绘制出分数阶导函数的三维图，如图 9-1 (b) 所示。

```
>> Z=[]; t=0:0.01:pi; y=exp(-t).*sin(3*t+1);
   for gam=0:0.1:1, Z=[Z; glfdiff(y,t,gam)]; end
   surf(t,0:0.1:1,Z); axis([0,pi,0,1,-1.2,6])
```



(a) 不同采样周期的结果比较
(b) 分数阶微分曲面

图 9-1 函数的分数阶微分

3. Riemann-Liouville 分数阶微积分公式

Riemann-Liouville 分数阶积分公式的定义为

$$
{}_a\mathscr{D}_t^{-\alpha}f(t)=\frac{1}{\Gamma(\alpha)}\int_a^t(t-\tau)^{\alpha-1}f(\tau)\mathrm{d}\tau \tag{9-1-6}
$$

其中，$0<\alpha<1$，且 a 为初值，一般可以假设零初始条件，即令 $a=0$，这时微分记号可以简写成 $\mathscr{D}_t^{-\alpha}f(t)$。Riemann-Liouville 定义为目前最常用的分数阶微积分定义。特别地，$\mathscr{D}$ 左右侧的下标分别表示积分式的下界和上界[4]。

由这样的积分还可以定义出分数阶微分。假设分数阶 $n-1<\beta\leqslant n$，则定义其分数阶微分为

$$
{}_a\mathscr{D}_t^{\beta}f(t)=\frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[{}_a\mathscr{D}_t^{-(n-\beta)}f(t)\right]=\frac{1}{\Gamma(n-\beta)}\frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[\int_a^t\frac{f(\tau)}{(t-\tau)^{\beta-n+1}}\mathrm{d}\tau\right] \tag{9-1-7}
$$

4. Caputo 分数阶微分定义

Caputo 分数阶微分定义为

$$
{}_0\mathscr{D}_t^{\alpha}y(t)=\frac{1}{\Gamma(1-\gamma)}\int_0^t\frac{y^{(m+1)}(\tau)}{(t-\tau)^{\gamma}}\mathrm{d}\tau \tag{9-1-8}
$$

其中，$\alpha=m+\gamma$, m 为整数，$0<\gamma\leqslant 1$。类似地，Caputo 分数阶积分定义为

$$
{}_0\mathscr{D}_t^{\gamma}=\frac{1}{\Gamma(-\gamma)}\int_0^t\frac{y(\tau)}{(t-\tau)^{1+\gamma}}\mathrm{d}\tau,\ \ \gamma<0 \tag{9-1-9}
$$

可以证明[2]，对很广一类实际函数来说，Grünwald-Letnikov 分数阶微积分定义及 Riemann-Liouville 分数阶微积分定义是完全等效的。Caputo 定义和 Riemann-Liouville 定义的区别主要表现在对常数求导的定义上，前者对常数的求导是有界的 (为 0)，而后者求导是无界的，Caputo 定义更适用于分数阶微分方程初值问题的描述。本节主要研究 Grünwald-Letnikov 分数阶微积分及其在控制中的应用问题。

例 9-2 试用 Grünwald-Letnikov 定义和 Cauchy 积分公式分别求取已知函数 $f(t)=\sin(3t+1)$ 的 0.75 阶微分，并比较得出的结果。

求解 由 Cauchy 定义，按式 (9-1-2) 给出的分数阶微分公式可以立即求出 0.75 阶微分为 ${}_0\mathscr{D}_t^{0.75}f(t)=3^{0.75}\sin\left(3t+1+\dfrac{0.75\pi}{2}\right)$，而用 Grünwald-Letnikov 定义的微分可以由 `glfdiff()` 函数得出。这样，由下面语句可以绘制出由这两个定义得出的分数阶微分曲线，如图 9-2 所示。

```
>> t=0:0.01:pi; y=sin(3*t+1); y1=3^0.75*sin(3*t+1+0.75*pi/2);
   y2=glfdiff(y,t,0.75); plot(t,y1,t,y2,'--')
```

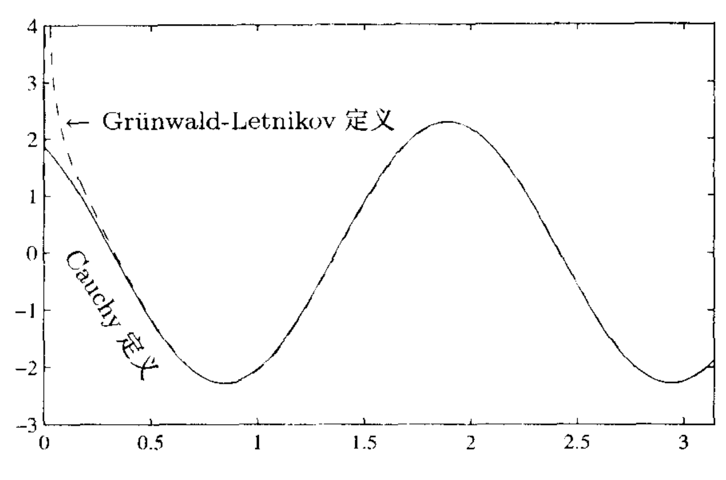


图 9-2 不同定义下的分数阶微分曲线

比较两种定义得出的结果，可见用 Cauchy 积分公式定义的算法没有初始突变过程，在 $t=0$ 区域以外二者几乎是一致的。从得出的结果很难断定哪个定义是正确的，因为这完全取决于 $t \leqslant 0$ 时 y 的值是什么。若 $y=0$，则显然在 $t=0^+$ 时刻 y 的值从 0 突变到 $\sin 1$，所以这时分数阶微分值应该为 ∞，突变的影响亦将持续一段时间，故 Grünwald-Letnikov 定义能正确反映该函数的分数阶微分变化，而不适合用 Cauchy 积分公式；若在 $t \leqslant 0$ 时原函数仍然满足函数 $y(t)=\sin(3t+1)$，则在 $t=0^+$ 时函数没有突变，这时更适合使用 Cauchy 积分公式。

9.1.2 分数阶微积分的性质

分数阶微积分有如下性质[5]：

① 解析函数 $f(t)$ 的分数阶导数 ${}_0\mathscr{D}_t^\alpha f(t)$ 对 t 和 α 都是解析的。

② $\alpha=n$ 为整数时，分数阶微分与整数阶完全一致，且 ${}_0\mathscr{D}_t^0 f(t)=f(t)$。

③ 分数阶微积分算子为线性的，即对任意常数 a,b，有

$$ {}_0\mathscr{D}_t^\alpha\left[af(t)+bg(t)\right]=a\,{}_0\mathscr{D}_t^\alpha f(t)+b\,{}_0\mathscr{D}_t^\alpha g(t) \tag{9-1-10} $$

④ 分数阶微积分算子满足交换律，并满足叠加关系

$$ {}_0\mathscr{D}_t^\alpha\left[{}_0\mathscr{D}_t^\beta f(t)\right]={}_0\mathscr{D}_t^\beta\left[{}_0\mathscr{D}_t^\alpha f(t)\right]={}_0\mathscr{D}_t^{\alpha+\beta} f(t) \tag{9-1-11} $$

⑤ 函数分数阶微分的 Laplace 变换为

$$ \mathscr{L}\left[{}_0\mathscr{D}_t^\alpha f(t)\right]=s^\alpha\mathscr{L}[f(t)]-\sum_{k=1}^{n-1}s^k\left[{}_0\mathscr{D}_t^{\alpha-k-1} f(t)\right]_{t=0} \tag{9-1-12} $$

特别地，若函数 $f(t)$ 及其各阶导数的初值均为 0，则 $\mathscr{L}[{}_0\mathscr{D}_t^\alpha f(t)]=s^\alpha\mathscr{L}[f(t)]$。

9.2 线性分数阶系统的时域与频域分析

分数阶系统是传统的整数阶系统的直接拓展。利用分数阶微积分的 Laplace 变换性质，分数阶系统也可以由传递函数模型描述，不同的是，这样的传递函数是分数阶的传递函数模型。例如，单变量系统的分数阶传递函数表示为

$$G(s)=\frac{b_1s^{\gamma_1}+b_2s^{\gamma_2}+\cdots+b_ms^{\gamma_m}}{a_1s^{\eta_1}+a_2s^{\eta_2}+\cdots+a_{n-1}s^{\eta_{n-1}}+a_ns^{\eta_n}} \tag{9-2-1}$$

其中分子分母多项式的系数 b_i,a_i 为实数，且阶次 γ_i,η_i 也为实数。分数阶传递函数的 Laplace 变换求解是很复杂的，对一般问题来说，得出该变换解析解是不可能的。本节将研究分数阶传递函数的时域与频域分析方法，并介绍建立分数阶传递函数 MATLAB 类的基本方法及应用。

9.2.1 分数阶传递函数模型的 MATLAB 描述

显然，如果给出式 (9-2-1) 中给出的分数阶传递函数模型的系数与阶次，可以惟一确定该传递函数模型。基于这样的考虑，可以用下面介绍的方法建立其一个分数阶传递函数的类 `fotf`。建立类的最基本步骤是建立一个 `@fotf` 目录，在目录中编写两个函数 `fotf()` 和 `display()`，前者描述该类的内容，后者描述该类的显示方法。具体地，可以首先建立起 `fotf.m` 文件：

```
function G=fotf(a,na,b,nb)
if nargin==0,
   G.a=[]; G.na=[]; G.b=[]; G.nb=[]; G=class(G,'fotf');
elseif isa(a,'fotf'), G=a;
elseif nargin==1 & isa(a,'double'), G=fotf(1,0,a,0);
else,
   ii=find(abs(a)<eps); a(ii)=[]; na(ii)=[];
   ii=find(abs(b)<eps); b(ii)=[]; nb(ii)=[];
   G.a=a; G.na=na; G.b=b; G.nb=nb; G=class(G,'fotf');
end
```

该函数的调用格式为 G=`fotf`$(\boldsymbol{a},\boldsymbol{\eta},\boldsymbol{b},\boldsymbol{\gamma})$，其中，向量 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 分别为系统的分母和分子多项式系数向量，而 $\boldsymbol{\eta}$ 和 γ 为系统的分母和分子的阶次向量。

另一个基本的函数为类显示函数 `display()`，其内容为

```
function display(G)
strN=polydisp(G.b,G.nb); strD=polydisp(G.a,G.na);
nm=max([length(strN),length(strD)]);
nn=length(strN); nd=length(strD);
disp([char(' '*ones(1,floor((nm-nn)/2))) strN]),
disp(char('-'*ones(1,nm)));
disp([char(' '*ones(1,floor((nm-nd)/2))) strD])
```

```
function strP=polydisp(p,np)
P=''; [np,ii]=sort(np,'descend'); p=p(ii);
for i=1:length(p),
   P=[P,'+',num2str(p(i)),'s^{',num2str(np(i)),'}'];
end
P=P(2:end); P=strrep(P,'s^{0}',''); P=strrep(P,'+-','-');
P=strrep(P,'^{1}',''); P=strrep(P,'+1s','+s');
strP=strrep(P,'-1s','-s');
if length(strP)>=2 & strP(1:2)=='1s', strP=strP(2:end); end
```

有了这两个函数，就可以在 MATLAB 工作空间中直接建立起一个分数阶传递函数类，然后在 MATLAB 环境中对该系统进行进一步分析，就像前面介绍的 tf 类一样。

例 9-3 试将分数阶传递函数输入到 MATLAB 工作空间。

$$G(s)=\frac{-2s^{0.63}-4}{2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5}$$

求解 先分别定义出系统的系数与阶次向量，则可以由下面的语句直接在 MATLAB 环境中建立起分数阶传递函数对象

```
>> b=[-2,-4]; nb=[0.63,0]; a=[2 3.8 2.6 2.5 1.5];
   na=[3.501,2.42,1.798,1.31,0]; G=fotf(a,na,b,nb)
```

由该对象内部的 display() 函数可以自动地按照规定格式将其显示出来。

还可以在 @tf 目录中建立起一个 fotf.m 文件，内容如下。该文件可以将一个整数阶传递函数模型对象转换成 fotf 对象。

```
function G1=fotf(G)
n=G.num{1}; d=G.den{1}; i1=find(abs(n)<eps); i2=find(abs(d)<eps);
if length(i1)>0 & i1(1)==1, n=n(i1(1)+1:end); end
if length(i2)>0 & i2(1)==1, d=d(i2(1)+1:end); end
G1=fotf(d,length(d)-1:-1:0,n,length(n)-1:-1:0);
```

9.2.2 分数阶模块的互联定义

前面介绍过整数阶系统模型的串联、并联和反馈连接。类似于这些连接方法，可以在 @fotf 目录下编写 plus.m, mtimes.m 和 feedback.m 文件，分别表示模块的串联、并联和反馈连接。这些函数使用了面向对象编程的重载技术。这些函数的内容分别为

① **加法函数** plus() 可以实现分数阶模块的并联

```
function G=plus(G1,G2)
a=kron(G1.a,G2.a); b=[kron(G1.a,G2.b), kron(G1.b,G2.a)];
```

```
na=[]; nb=[];
for i=1:length(G1.a),
    na=[na G1.na(i)+G2.na]; nb=[nb, G1.na(i)+G2.nb];
end
for i=1:length(G1.b), nb=[nb G1.nb(i)+G2.na]; end
G=unique(fotf(a,na,b,nb));
```

② **乘法函数** mtimes() 可以实现模块的串联

```
function G=mtimes(G1,G2)
G2=fotf(G2); a=kron(G1.a,G2.a);
b=kron(G1.b,G2.b); na=[]; nb=[];
for i=1:length(G1.na), na=[na,G1.na(i)+G2.na]; end
for i=1:length(G1.nb), nb=[nb,G1.nb(i)+G2.nb]; end
G=unique(fotf(a,na,b,nb));
```

③ **反馈函数** feedback() 可以实现模块的反馈连接

```
function G=feedback(F,H)
H=fotf(H); b=kron(F.b,H.a);
a=[kron(F.b,H.b), kron(F.a,H.a)]; na=[]; nb=[];
for i=1:length(F.b),
   nb=[nb F.nb(i)+H.nb]; na=[na,F.nb(i)+H.nb];
end
for i=1:length(F.a), na=[na F.na(i)+H.na]; end
G=unique(fotf(a,na,b,nb));
```

④ **化简函数** unique(): 因为通过前面的连接函数可能出现可以对消的零极点，或可能出现同类项，所以可以定义出下面的函数，对得出的分数阶传递函数对象进行化简

```
function G=unique(G)
[a,n]=polyuniq(G.a,G.na); G.a=a; G.na=n;
[a,n]=polyuniq(G.b,G.nb); G.b=a; G.nb=n;
function [a,an]=polyuniq(a,an)
[an,ii]=sort(an,'descend'); a=a(ii); ax=diff(an); key=1;
for i=1:length(ax)
   if ax(i)==0,
      a(key)=a(key)+a(key+1); a(key+1)=[]; an(key+1)=[];
   else, key=key+1; end
end
```

⑤ **自减函数** uminus(): 定义了自减函数就可以计算 $-G$

```
function G=uminus(G1)
G=fotf(G1.a,G1.na,-G1.b,G1.nb);
```

⑥ **减法函数**: 定义了减法函数就可以计算 $G_1 - G_2$

```
function G=minus(G1,G2)
G=G1+(-G2);
```

⑦ **逆函数** inv()：可以计算 $1/G$

```
function G=inv(G1)
G=fotf(G1.b,G1.nb,G1.a,G1.na);
```

例 9-4 假设单位负反馈系统的分数阶子模型为

$$G(s)=\frac{0.8s^{1.2}+2}{1.1s^{1.8}+0.8s^{1.3}+1.9s^{0.5}+0.4},\ G_{\rm c}(s)=\frac{1.2s^{0.72}+1.5s^{0.33}}{3s^{0.8}}$$

试求出闭环系统的传递函数模型。

求解 可以先单独输入系统的对象模型和控制器模型，然后用下面的命令直接得出闭环系统的分数阶传递函数模型

```
>> G=fotf([1.1,0.8 1.9 0.4],[1.8 1.3 0.5 0],[0.8 2],[1.2 0]);
   Gc=fotf([3],[0.8],[1.2 1.5],[0.72 0.33]); Ga=feedback(G*Gc,1)
```

这样得出的闭环模型为

$$G_{\rm a}(s)=\frac{0.96s^{1.92}+1.2s^{1.53}+2.4s^{0.72}+3s^{0.33}}{3.3s^{2.6}+2.4s^{2.1}+0.96s^{1.92}+1.2s^{1.53}+5.7s^{1.3}+1.2s^{0.8}+2.4s^{0.72}+3s^{0.33}}$$

可见，通过上面几个语句就可以得出系统的闭环模型，所以这样定义的函数使用起来还是很方便的，它可以化简整个分数阶系统的分析与设计过程。

9.2.3 分数阶系统的频域分析

对分数阶传递函数来说，如果用 $\mathrm{j}\omega$ 直接取代传递函数中的 s，则可以将分数阶传递函数闭环成和 ω 相关的 $G(\mathrm{j}\omega)$ 函数，该函数可以同样分别写成实部和虚部的表示形式或幅值相位的表示形式，这样可以简单地得出分数阶系统的频域响应数据。根据这样得出的频域响应数据，可以同样绘制出系统的 Bode 图、Nyquist 图和 Nichols 图。可以在 @fotf 目录下的重载函数 bode.m 文件绘制出系统的 Bode 图，该函数的内容为

```
function H=bode(G,w)
a=G.a; eta=G.na; b=G.b; gamma=G.nb;
if nargin==1, w=logspace(-4,4); end
for i=1:length(w)
    P=b*((sqrt(-1)*w(i)).^gamma.');
    Q=a*((sqrt(-1)*w(i)).^eta.'); H1(i)=P/Q;
end
H1=frd(H1,w); if nargout==0, bode(H1); else, H=H1; end
```

该函数的调用格式为 H=bode(G,ω)，其中 G 为分数阶传递函数对象，ω 为用户选定的频率向量，该变量可以省略，采用默认值。和整数阶系统的 bode() 函数类似，若不返回 H 变量，则会自动绘制出系统的 Bode 图。类似地，在该目录中还可以编写 nyquist.m 文件和 nichols.m 文件。

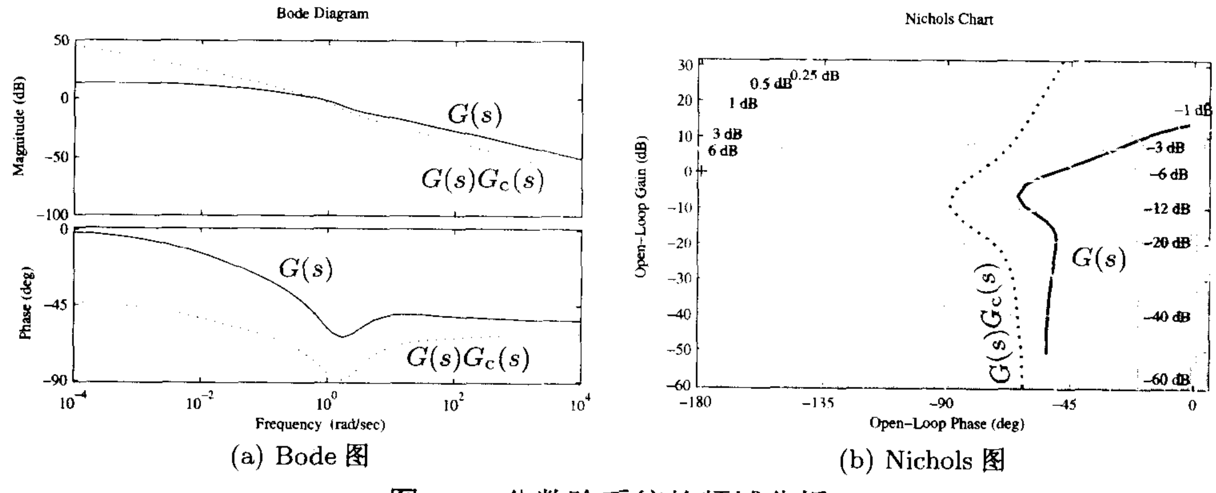
```
function nyquist(G,w)
if nargin==1, w=logspace(-4,4); end; H=bode(G,w); nyquist(H);

function nichols(G,w)
if nargin==1, w=logspace(-4,4); end; H=bode(G,w); nichols(H);
```

例 9-5 试绘制出例 9-4 中给出的开环模型 Bode 图和 Nichols 图。

求解 可以由下面语句分别绘制出校正前与校正后的开环系统 Bode 图和 Nichols 图，如图 9-3 (a)、(b) 所示。可见，使用分数阶传递函数对象后，系统分析几乎和整数阶系统一样容易。

```
>> G=fotf([1.1,0.8 1.9 0.4],[1.8 1.3 0.5 0],[0.8 2],[1.2 0]);
   Gc=fotf([3],[0.8], [1.2 1.5],[0.72 0.33]); H=fotf(1,0,1,0);
   H1=bode(G); H2=bode(G*Gc); bode(H1,H2,':')
   figure; nichols(H1,H2,':'); grid
```



(a) Bode 图　　(b) Nichols 图

图 9-3 分数阶系统的频域分析

9.2.4 分数阶系统的时域分析

和频域响应分析相比，分数阶系统的时域分析要复杂一些。考虑一类简单的分数阶微分方程[2]

$$a_1\mathscr{D}_t^{\eta_1}y(t)+a_2\mathscr{D}_t^{\eta_2}y(t)+\cdots+a_{n-1}\mathscr{D}_t^{\eta_{n-1}}y(t)+a_n\mathscr{D}_t^{\eta_n}y(t)=u(t) \qquad (9\text{-}2\text{-}2)$$

其中 $u(t)$ 为某已知函数。假设输出信号 $y(t)$ 及其各阶导数的初值均为 0，则可以

由 Laplace 变换写出系统的传递函数模型

$$G(s)=\frac{1}{a_1s^{\eta_1}+a_2s^{\eta_2}+\cdots+a_{n-1}s^{\eta_{n-1}}+a_ns^{\eta_n}} \tag{9-2-3}$$

再考虑 Grünwald-Letnikov 分数阶微积分定义，可以写出上述微分方程的离散形式为

$${}_a\mathscr{D}_t^{\eta_i}y(t)\approx\frac{1}{h^{\eta_i}}\sum_{j=0}^{[(t-a)/h]}w_j^{(\eta_i)}y(t-jh)=\frac{1}{h^{\eta_i}}\left[y(t)+\sum_{j=1}^{[(t-a)/h]}w_j^{(\eta_i)}y(t-jh)\right] \tag{9-2-4}$$

其中 $w_0^{(\beta_i)}$ 可以由式 (9-1-4) 递推求出。代入式 (9-2-2)，则可以写出分数阶微分方程的数值解为

$$y(t)=\frac{1}{\displaystyle\sum_{i=1}^{n}\frac{a_i}{h^{\eta_i}}}\left[u(t)-\sum_{i=1}^{n}\frac{a_i}{h^{\eta_i}}\sum_{j=1}^{[(t-a)/h]}w_j^{(\eta_i)}y(t-jh)\right] \tag{9-2-5}$$

现在考虑式 (9-2-1) 中给出的一般形式。如果先对等号右侧的函数 $u(t)$ 求分数阶导数，则显然可以将原方程变换成右侧为 $\hat{u}(t)$ 的形式，这样套用上述的公式就可以求出一般微分方程的数值解。在实际编程运算中，先求导可能导致计算误差，故可以考虑先求在 $u(t)$ 激励下的 $\hat{y}(t)$，再对得出的 $\hat{y}(t)$ 按照等号右侧的方式求导。对线性系统来说这个方法是完全等效的。基于这个算法，可以编写出一个重载函数 `lsim()` 来实现任意输入的分数阶系统数值解法。

```
function y=lsim(G,u,t)
a=G.a; eta=G.na; b=G.b; gamma=G.nb; nA=length(a); h=t(2)-t(1);
D=sum(a./[h.^eta]); W=[]; nT=length(t); vec=[eta gamma];
D1=b(:)./h.^gamma(:); y1=zeros(nT,1); W=ones(nT,length(vec));
for j=2:nT, W(j,:)=W(j-1,:).*(1-(vec+1)/(j-1)); end
for i=2:nT
    A=[y1(i-1:-1:1)]'*W(2:i,1:nA);
    y1(i)=(u(i)-sum(A.*a./[h.^eta]))/D;
end
for i=2:nT, y(i)=(W(1:i,nA+1:end)*D1)'*[y1(i:-1:1)]; end
```

该函数的调用格式为 **y=lsim(*G*,*u*,*t*)** ，其中时间向量和输入点向量分别由 $\boldsymbol{t}$ 和 $\boldsymbol{u}$ 给出。注意，当计算点需要很多时，这样的求解方法可能比较慢。

基于 `lsim()` 函数，可以编写一个重载的 `step()` 函数来绘制分数阶系统的阶跃响应曲线

```
function y=step(G,t)
```

```
u=ones(size(t)); y=lsim(G,u,t); if nargout==0, plot(t,y); end
```

该函数的调用格式为 **y=step(G,t)**，其中 G 为分数阶传递函数模型，t 为时间向量，系统的阶跃响应数据由向量 y 返回。若调用函数时不返回任何变量，则将自动绘制阶跃响应曲线。

例 9-6 试绘制出分数阶系统 $G(s)=\dfrac{5}{s^{0.8}+0.75s^{0.4}+0.9}$ 的阶跃响应曲线。

求解 选择不同的步长，由下面语句可以立即绘制出系统的阶跃响应曲线，如图 9-4 所示。可见，如果不是将步长选择太大，如本例中的 0.1，step() 函数还是可以精确地求出系统的时域响应的。值得指出的是，在仿真结束后应该选择另外一个不同的步长来检验仿真结果是否正确。

```
>> G=fotf([1 0.75 0.9],[0.8 0.4 0],5,0);
   t1=0:0.001:10; t2=0:0.01:10; t3=0:0.1:5;
   y1=step(G,t1); y2=step(G,t2); y3=step(G,t3);
   plot(t1,y1,t2,y2,'--',t3,y3,':')
```

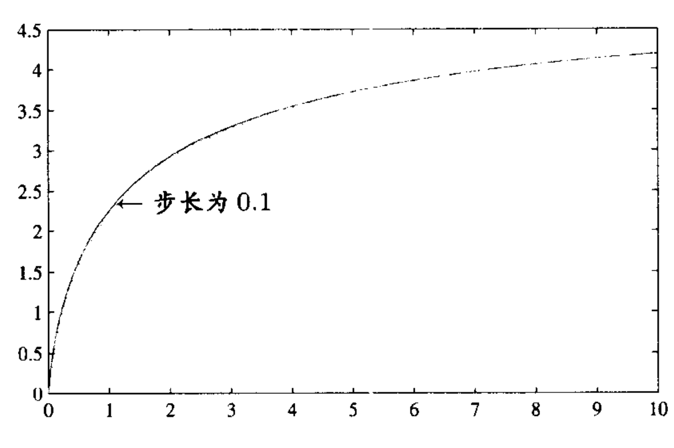


图 9-4 分数阶系统的阶跃响应曲线

9.2.5 一类分数阶线性系统时域响应解析解方法

类似于整数阶函数的部分分式展开法，求解一类线性系统时域响应解析解可以通过引入 Mittag-Leffler 函数来获得。Mittag-Leffler 函数是指数函数 e^z 的拓广。带有两个参数的 Mittag-Leffler 函数的定义为

$$\mathscr{E}_{\alpha,\beta}(z)=\sum_{k=0}^{\infty}\frac{z^k}{\Gamma(\alpha k+\beta)},\quad (\alpha,\beta>0) \tag{9-2-6}$$

显然，指数函数 e^z 是 Mittag-Leffler 函数的一个特例

$$\mathscr{E}_{1,1}(z)=\sum_{k=0}^{\infty}\frac{z^k}{\Gamma(k+1)}=\sum_{k=0}^{\infty}\frac{z^k}{k!}=\mathrm{e}^z$$

由 Mittag-Leffler 函数的定义还可以导出其他的特例，如

$$\mathscr{E}_{2,1}(z)=\cosh(\sqrt{z}),\ \mathscr{E}_{1,2}(z)=\frac{\mathrm{e}^{z}-1}{z},\ \mathscr{E}_{2,2}(z)=\frac{\sinh(\sqrt{z})}{\sqrt{z}} \tag{9-2-7}$$

$$\mathscr{E}_{1/2,1}(\sqrt{z})=\frac{2}{\sqrt{\pi}}\mathrm{e}^{-z}\mathrm{erfc}(-\sqrt{z}) \tag{9-2-8}$$

这样，由 n 项构成的分数阶微分方程的解可以表示为

$$\begin{aligned} y(t) \ =\ & \frac{1}{a_n}\sum_{m=0}^{\infty}\frac{(-1)^m}{m!}\sum_{\substack{k_0+k_1+\cdots+k_{n-2}=m\\ k_0\geqslant 0,\ \cdots,\ k_{n-2}\geqslant 0}}(m;k_0,k_1,\cdots,k_{n-2}) \\ & \prod_{i=0}^{n-2}\left(\frac{a_i}{a_n}\right)^{k_i} t^{(\beta_n-\beta_{n-1})m+\beta_n+\sum\limits_{j=0}^{n-2}(\beta_{n-1}-\beta_j)k_j-1} \\ & \mathscr{E}^{(m)}_{\beta_n-\beta_{n-1},\ \beta_n+\sum\limits_{j=0}^{n-2}(\beta_{n-1}-\beta_j)k_j}\left(-\frac{a_{n-1}}{a_n}t^{\beta_n-\beta_{n-1}}\right) \end{aligned} \tag{9-2-9}$$

式中 $\mathscr{E}_{\alpha,\beta}(x)$ 为式 (9-2-6) 中定义的两参数 Mittag-Leffler 函数，且

$$\mathscr{E}^{(n)}_{\alpha,\beta}(x)=\frac{\mathrm{d}^n}{\mathrm{d}x^n}\mathscr{E}_{\alpha,\beta}(x)=\sum_{j=0}^{\infty}\frac{(j+n)!\ x^j}{j!\ \Gamma(\alpha j+\alpha n+\beta)} \tag{9-2-10}$$

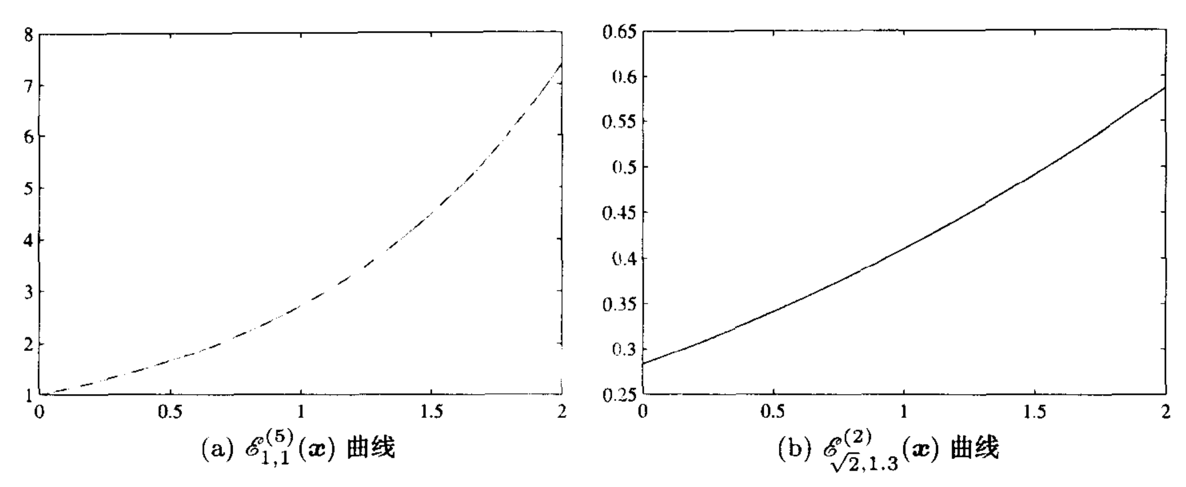
其中 $n=0,1,2,\cdots$。用 MATLAB 语言可以立即编写出求取 Mittag-Leffler 函数的 n 阶导函数的函数 ml_fun.m

```
function f=ml_fun(a,b,x,n,eps0)
if nargin<5, eps0=eps; end
if nargin<4, n=0; end, f=0; fa=1; j=0;
while norm(fa,1)>=eps0
    fa=gamma(j+n+1)/gamma(j+1)/gamma(a*j+a*n+b)*x.^j;
    f=f+fa; j=j+1;
end
```

该函数的调用格式为 $\boldsymbol{f}$=ml_fun(α,β,$\boldsymbol{y}$,n,ϵ_0)，其中 ϵ_0 为可以接受的误差限。得出的 $\boldsymbol{f}$ 向量即为所需的 $\mathscr{E}^{(n)}_{\alpha,\beta}(\boldsymbol{y})$，其中 $\boldsymbol{y}$ 为自变量向量。函数后两个变量的默认值为 $n=0$，ϵ_0 =eps。

例 9-7 分别绘制出 $\mathscr{E}^{(5)}_{1,1}(\boldsymbol{y})$ 和 $\mathscr{E}^{(2)}_{\sqrt{2},1.3}(\boldsymbol{y})$ 曲线。

求解 由前面的叙述已知，前者的解析解为 e^{y}，而后者没有解析解。下面的语句可以绘制出这两个函数的曲线，如图 9-5 (a)、(b) 所示，其中解析解的函数曲线和计算出来的结果是完全重合的。



(a) $\mathscr{E}_{1,1}^{(5)}(\boldsymbol{x})$ 曲线　　(b) $\mathscr{E}_{\sqrt{2},1.3}^{(2)}(\boldsymbol{x})$ 曲线

图 9-5　Mittag-Leffler 导函数曲线

```
>> x=0:0.001:2; y1=ml_fun(1,1,x,5); plot(x,y1,x,exp(x),'--')
   figure; y2=ml_fun(sqrt(2),1.3,x,2); plot(x,y2)
```

一般系统的时域响应解析解表达式 (9-2-9) 有时过于复杂，所以这里只考虑一个特殊模型 $G(s)=\dfrac{1}{a_2s^{\beta_2}+a_1s^{\beta_1}+a_0}$ 的阶跃响应解 [6]

$$y(t)=\frac{1}{a_2}\sum_{k=0}^{\infty}\frac{(-1)^k\hat{a}_0^k t^{-\hat{a}_1+(k+1)\beta_2}}{k!}\mathscr{E}_{\beta_2-\beta_1,\beta_2+\beta_1k+1}^{(k)}\left(-\hat{a}_1t^{\beta_2-\beta_1}\right) \tag{9-2-11}$$

其中 $\hat{a}_0=a_0/a_2,\hat{a}_1=a_1/a_2$。仿照前面的 `ml_fun()` 函数可以编写出下面的阶跃响应数值求解函数

```
function y=ml_step(a0,a1,a2,b1,b2,t,eps0)
y=0; k=0; ya=1; a0=a0/a2; a1=a1/a2;
if nargin==6, eps0=eps; end
while max(abs(ya))>=eps0
    ya=(-1)^k/gamma(k+1)*a0^k*t.^((k+1)*b2).*...
        ml_fun(b2-b1,b2+b1*k+1,-a1*t.^(b2-b1),k,eps0);
    y=y+ya; k=k+1;
end
y=y/a2;
```

该函数的调用格式为 $\boldsymbol{y}$=`ml_step`$(a_0,a_1,a_2,\beta_1,\beta_2,\boldsymbol{t},\epsilon_0)$ 。

例 9-8 重新考虑例 9-6 中给出的系统，试用 Mittag-Leffler 函数的形式求解该系统的阶跃响应曲线，并比较前面得出的结果。

求解 显然，$a_0=0.9,a_1=0.75,a_2=1,\beta_1=0.4,\beta_2=0.8$，这样由下面的语句可以得出原系统阶跃响应的数值解，和前面由 step() 函数得出的曲线和图 9-4 给出的完全一致，无法区分。二者之间的误差如图 9-6 所示，可见两种方法得出的都是很精确的。

```
>> t=0:0.001:5; y=ml_step(0.9,0.75,1,0.4,0.8,t);
   G=fotf([1 0.75 0.9],[0.8 0.4 0],5,0); y1=step(G,t);
   plot(t,5*y,t,y1,'--'), figure; plot(t,5*y-y1)
```

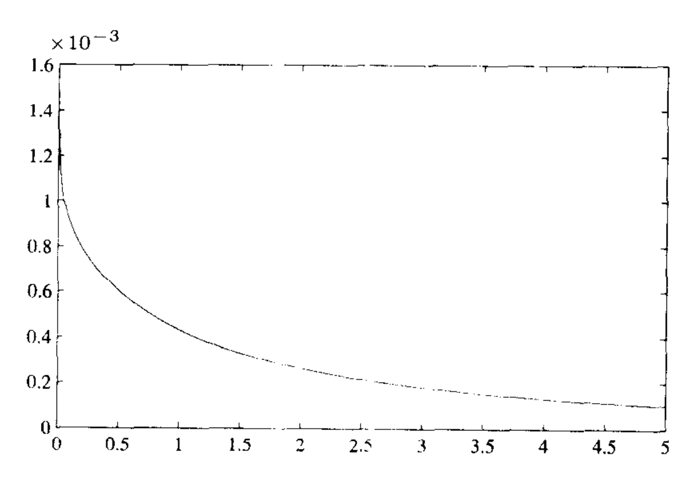


图 9-6 两种方法得出的阶跃响应曲线之间的误差曲线

从前面实际例子的仿真可见，由基于 Grünwald-Letnikov 定义的分数阶微分计算得出的阶跃响应计算量比 Mittag-Leffler 函数的算法计算量明显小，适用面广很多，所以在后面的研究中侧重于该方法的使用。

9.3 分数阶微分环节的滤波器近似

前面给出了各种分数阶微分的定义并着重研究了 Grünwald-Letnikov 微分算法，然而，这些定义能够使用的前提是，函数 $y(t)$ 的表达式是已知的，$f(t)$ 未知，则这些定义无法使用。在控制系统研究中，系统内部信号的解析表达式通常是不知道的，所以需要某种方法直接由信号的采样点来近似信号的分数阶微积分。这里将介绍的基于滤波器的计算方法是解决这一问题的有效方法。

9.3.1 Oustaloup 递推滤波器

文献 [5] 总结了对分数阶微分器的各种连续的滤波器逼近，离散滤波器逼近也有相应的研究[7]，在众多滤波器中，Oustaloup 递推滤波器[8]的性能是很出众的。由于纯微分器的频域特性是斜线，所以不存在任何滤波器能在全频段对微分器有很好的近似，只能在某个感兴趣频段内实现微分器的逼近。假设需要逼近的频段为 $(\omega_\mathrm{b},\omega_\mathrm{h})$，可以将相应的滤波器写成

$$G_\mathrm{f}(s)=K\prod_{k=-N}^{N}\frac{s+\omega_k'}{s+\omega_k} \tag{9-3-1}$$

其中，零极点和增益可以由式 (9-3-2) 计算出来

$$\omega'_k = \omega_b \left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1-\gamma)}{2N+1}}, \quad \omega_k = \omega_b \left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1+\gamma)}{2N+1}}, \quad K = \omega_h^{\gamma} \tag{9-3-2}$$

根据上述的算法可以直接编写出 MATLAB 函数 oustafod() 来构造连续整数阶滤波器。这时当信号 $y(t)$ 通过此滤波器后，滤波器的输出可以近似地看成 $\mathscr{D}_t^{\gamma} y(t)$ 信号的近似。

```
function G=oustafod(r,N,wb,wh)
mu=wh/wb; k=-N:N; w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1))*wb;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1))*wb;
K=wh^r; G=tf(zpk(-w_kp',-w_k',K));
```

该函数的调用格式为 G_f=oustafod(γ,N,ω_b,ω_h)，其中，γ 为微分的阶次；变量 $2N+1$ 为滤波器的阶次；(ω_b, ω_h) 为感兴趣的频段；返回的 G_f 即为设计出的滤波器模型。

例 9-9 假设 $\omega_b = 0.01, \omega_h = 100\,\text{rad/sec}$，试设计出连续滤波器，对 $f(t) = \mathrm{e}^{-t}\sin(3t+1)$ 信号计算 0.5 阶微分。

求解 选择感兴趣的频段为 $(0.01, 100)$，并选择阶次 $N = 2, 3$，则可以通过下面的语句直接设计出滤波器

```
>> G1=oustafod(0.5,2,0.01,100), G2=oustafod(0.5,3,0.01,100)
   bode(G1,G2)
```

这样设计的两个滤波器分别为

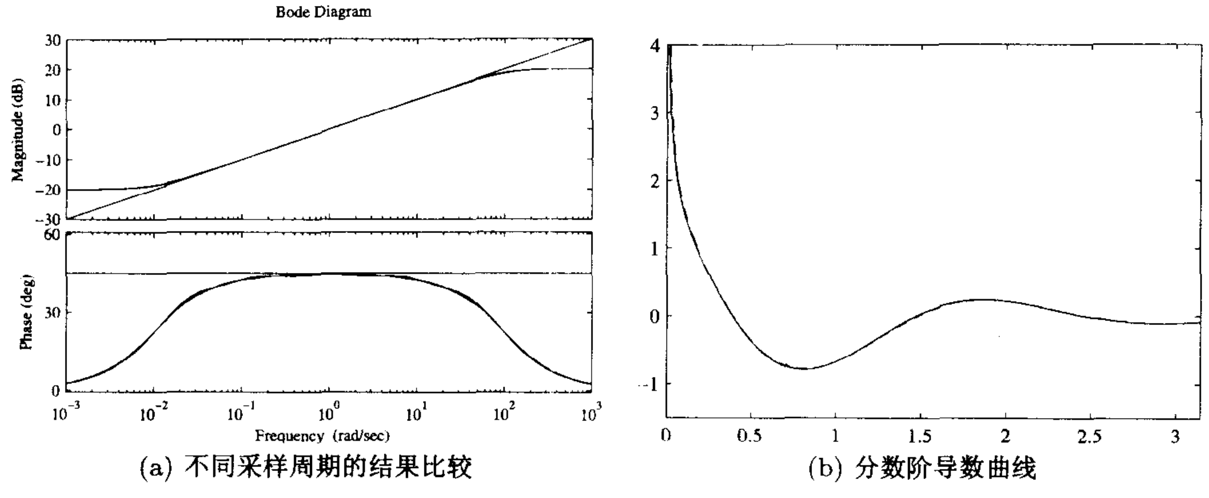
$$G_1(s) = \frac{10s^5 + 298.5s^4 + 1218s^3 + 768.5s^2 + 74.97s + 1}{s^5 + 74.97s^4 + 768.5s^3 + 1218s^2 + 298.5s + 10}$$

$$G_2(s) = \frac{10s^7 + 509.4s^6 + 5487s^5 + 1.499\times10^4 s^4 + 1.079\times10^4 s^3 + 2045s^2 + 98.34s + 1}{s^7 + 98.34s^6 + 2045s^5 + 1.079\times10^4 s^4 + 1.499\times10^4 s^3 + 5487s^2 + 509.4s + 10}$$

上面的语句还可以直接绘制出该滤波器的 Bode 图，如图 9-7 (a) 所示，在该图中还叠印了直线，表示 $(\mathrm{j}\omega)^{\gamma}$ 的理论值。由下面的语句还可以绘制出由滤波器计算出来的分数阶微分曲线，同时也可绘制出由 Grünwald-Letnikov 定义计算出来的分数阶微分曲线，如图 9-7 (b) 所示。可见，由滤波器计算出来的分数阶微分结果还是很精确的。

```
>> t=0:0..001:pi; y=exp(-t).*sin(3*t+1); y0=glfdiff(y,t,0.5);
   y1=lsim(G1,y,t); y2=lsim(G2,y,t); plot(t,y0,t,y1,'--',t,y2,':')
```

当然，用该算法还可以在更大的频率范围内拟合分数阶微分函数，这时需要适当增大拟合的阶次。下面给出在 $(10^{-4}, 10^4)$ 频段内的拟合效果，如图 9-8 所示。可见，对这样大的频率范围，不再适合 $N = 2$ 的近似，而应该采用更大的 N 值，如 $N = 4$。



(a) 不同采样周期的结果比较　　(b) 分数阶导数曲线

图 9-7　函数的分数阶导数

```
>> G=oustafod(0.5,2,1e-4,1e4); G1=oustafod(0.5,3,1e-4,1e4);
   G2=oustafod(0.5,4,1e-4,1e4); G3=oustafod(0.5,5,1e-4,1e4);
   bode(G,'-',G1,'--',G2,':',G3,'-.')
```

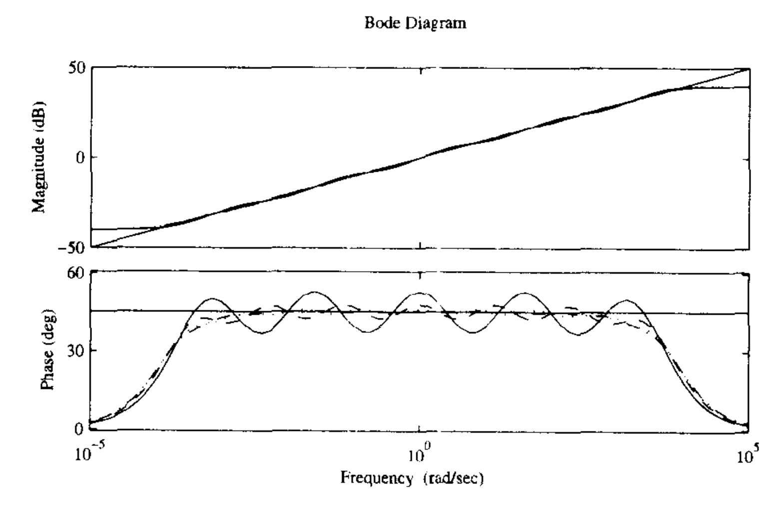


图 9-8　不同阶次下的滤波器近似效果

例 9-10 试用滤波器近似的方法求出分数阶传递函数模型的整数阶近似模型。

$$G(s)=\frac{-2s^{0.63}-4}{2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5}$$

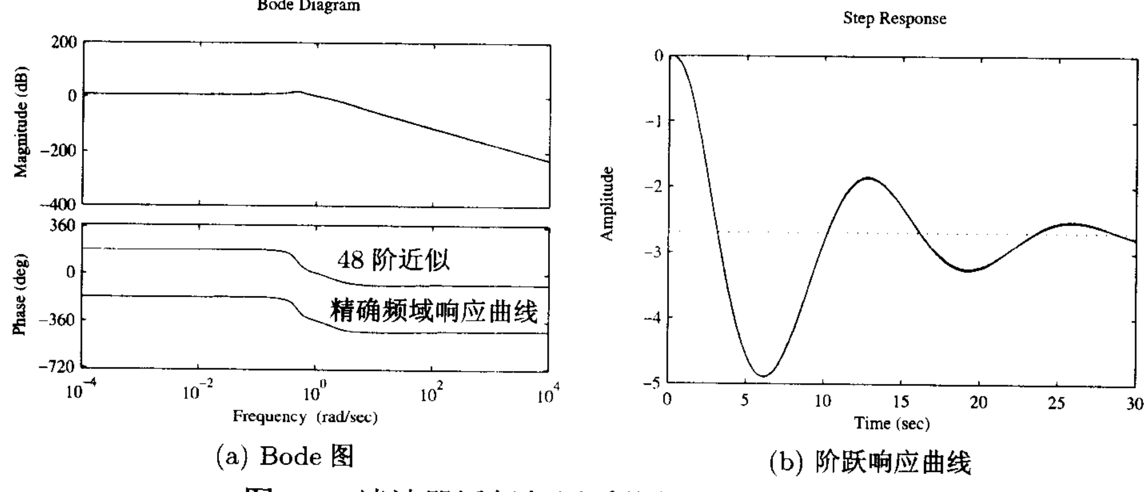
求解 可以考虑将传递函数中的分数阶微分表达式均由 Oustaloup 滤波器替换，这样就可以用整数阶传递函数逼近原始的分数阶传递函数了。假设感兴趣的频段为 $(10^{-3},10^4)$，高阶项如 $s^{3.501}$ 可以由 $s^3s^{0.501}$ 近似，这样就可以通过下面的语句逼近原分数阶微分方程模型了。

```
>> N=4; w1=1e-3; w2=1e4; g1=oustafod(0.501,N,w1,w2); s=tf('s');
   g2=oustafod(0.42,N,w1,w2); g3=oustafod(0.798,N,w1,w2);
```

```
g4=oustafod(0.31,N,w1,w2); g5=oustafod(0.63,N,w1,w2);
G1=(-2*g5-4)/(2*s^3*g1+3.8*s^2*g2+2.6*s*g3+2.5*s*g4+1.5);
```

可见，这样得出的整数阶近似的阶次是相当高的，达到 48 阶。下面的语句可以绘制出原系统和近似系统的 Bode 图和阶跃响应曲线，分别如图 9-9 (a)、(b) 所示。在 Bode 图的相频特性中二者似乎有很大差异，事实上，二者相差 360°，所以可以认为二者是完全一致的。由图中的响应曲线可见，这样得出的整数阶近似是相当精确的。

```
>> b=[-2 -4]; nb=[0.63 0]; a=[2 3.8 2.6 2.5 1.5];
   na=[3.501 2.42 1.798 1.31 0]; G=fotf(a,na,b,nb);
   w=logspace(-4,4,500); H=bode(G,w); bode(G1,H,{1e-4,1e4});
   figure; t=0:0.004:30; y=step(G,t); step(G1,30); line(t,y)
```



(a) Bode 图　　(b) 阶跃响应曲线

图 9-9　滤波器近似与原系统的时域、频域比较

9.3.2　改进的 Oustaloup 滤波器

从前面介绍的 Oustaloup 滤波器拟合效果可见，在频段 $[\omega_\mathrm{b},\omega_\mathrm{h}]$ 的两个端点附近的拟合效果不理想，所以这里将提出一个改进的滤波器方案[9]。在拟合的频段内，分数阶微分算子 s^α 可以由分数阶传递函数来近似

$$K(s)=\left(\frac{1+\dfrac{s}{\frac{d}{b}\omega_\mathrm{b}}}{1+\dfrac{s}{\frac{b}{d}\omega_\mathrm{h}}}\right)^\alpha \tag{9-3-3}$$

其中 $0<\alpha<1$，$s=j\omega$，$b>0$，$d>0$，且 $K(s)=\left(\dfrac{bs}{d\omega_\mathrm{b}}\right)^\alpha\left(1+\dfrac{-ds^2+d}{ds^2+b\omega_\mathrm{h}s}\right)^\alpha$。在频段 $\omega_\mathrm{b}<\omega<\omega_\mathrm{h}$，由 Taylor 幂级数可以表示为

$$K(s)=\left(\frac{bs}{d\omega_\mathrm{b}}\right)^\alpha\left(1+\alpha p(s)+\frac{\alpha(\alpha-1)}{2}p^2(s)+\cdots\right) \tag{9-3-4}$$

其中 $p(s)=\dfrac{-ds^2+d}{ds^2+b\omega_{\rm h}s}$。可以导出

$$s^\alpha=\frac{(d\omega_{\rm b})^\alpha b^{-\alpha}}{\left[1+\alpha p(s)+\dfrac{\alpha(\alpha-1)}{2}p^2(s)+\cdots\right]}\left(\frac{1+\dfrac{s}{\frac{d}{b}\omega_{\rm b}}}{1+\dfrac{s}{\frac{b}{d}\omega_{\rm h}}}\right)^\alpha \tag{9-3-5}$$

将 Taylor 级数截断到其第一项，则

$$s^\alpha\approx\frac{(d\omega_{\rm b})^\alpha}{b^\alpha\big(1+\alpha p(s)\big)}\left(\frac{1+\dfrac{s}{\frac{d}{b}\omega_{\rm b}}}{1+\dfrac{s}{\frac{b}{d}\omega_{\rm h}}}\right)^\alpha \tag{9-3-6}$$

这时，分数阶微分算子可以近似成

$$s^\alpha\approx\left(\frac{d\omega_{\rm b}}{b}\right)^\alpha\left(\frac{ds^2+b\omega_{\rm h}s}{d(1-\alpha)s^2+b\omega_{\rm h}s+d\alpha}\right)\left(\frac{1+\dfrac{s}{\frac{d}{b}\omega_{\rm b}}}{1+\dfrac{s}{\frac{b}{d}\omega_{\rm h}}}\right)^\alpha \tag{9-3-7}$$

可以证明，式 (9-3-7) 中给出的多项式在感兴趣频段 $(\omega_{\rm b},\omega_{\rm h})$ 内是稳定的[9]。这样，式 (9-3-7) 中的非有理部分可以近似成

$$K(s)=\lim_{N\to\infty}K_N(s)=\lim_{N\to\infty}\prod_{k=-N}^{N}\frac{1+s/\omega_k'}{1+s/\omega_k} \tag{9-3-8}$$

这样，由前面介绍的递推方法，可以写出分数阶微分算子的改进滤波器

$$s^\alpha\approx\left(\frac{d\omega_{\rm h}}{b}\right)^\alpha\left(\frac{ds^2+b\omega_{\rm h}s}{d(1-\alpha)s^2+b\omega_{\rm h}s+d\alpha}\right)\prod_{k=-N}^{N}\frac{s+\omega_k'}{s+\omega_k} \tag{9-3-9}$$

其中，实数极点和零点可以写成

$$\omega_k'=\left(\frac{d\omega_{\rm b}}{b}\right)^{\frac{\alpha-2k}{2N+1}},\quad \omega_k=\left(\frac{b\omega_{\rm h}}{d}\right)^{\frac{\alpha+2k}{2N+1}} \tag{9-3-10}$$

出于实际拟合效果考虑，一般可以取 $b=10$, $d=9$。基于上述算法，可以编写出 MATLAB 函数 new_fod() 来设计滤波器

```
function G=new_fod(r,N,wb,wh,b,d)
if nargin==4, b=10; d=9; end
```

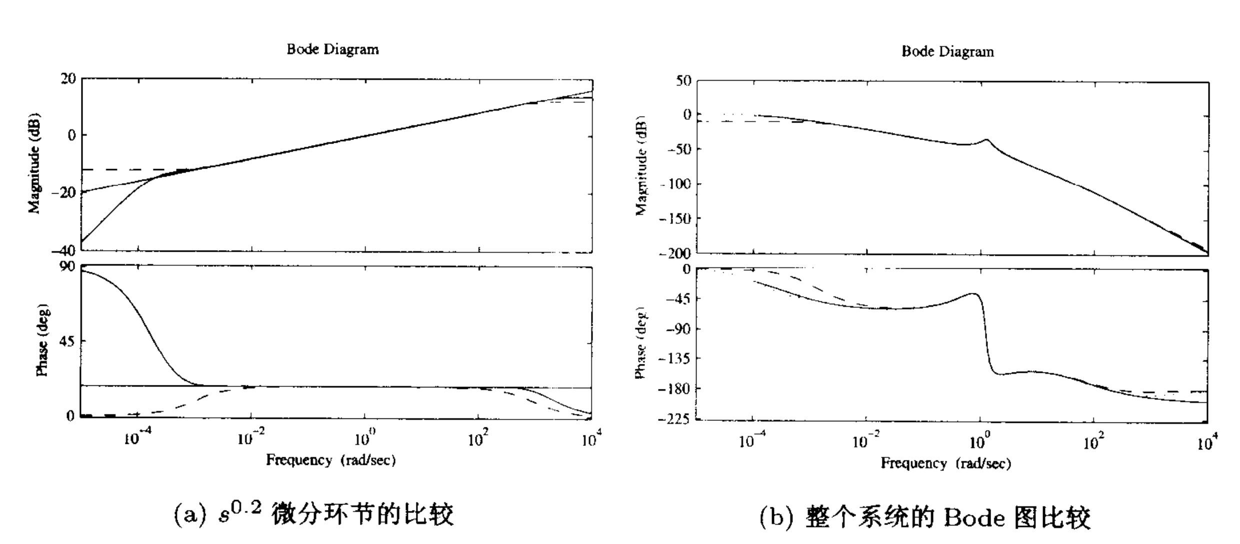
```
mu=wh/wb; k=-N:N; w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1))*wb;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1))*wb; K=(d*wh/b)^r;
G=zpk(-w_kp',-w_k',K)*tf([d,b*wh,0],[d*(1-r),b*wh,d*r]);
```

该函数的调用格式为 G_f=new_fod(γ,N,ω_b,ω_h,b,d) 。

例 9-11 考虑分数阶模型 $G(s)=\dfrac{s+1}{10s^{3.2}+185s^{2.5}+288s^{0.7}+1}$，试用 Oustaloup 滤波器和改进的滤波器逼近该模型，并比较逼近效果。

求解 由前面的介绍可知，分数阶线性系统的精确 Bode 图可以由重载的 bode() 函数直接绘制。由下面语句也可以容易地绘制出两种不同滤波器下整个系统的近似 Bode 图，对 0.2 阶微分的近似比较如图 9-10 (a) 所示，而对整个系统的近似比较如图 9-10 (b) 所示。可见，对纯微分的近似在整个感兴趣频段的拟合效果，尤其是在边界处，远远优于传统的 Oustalop 滤波器，而对整个系统的拟合效果优势更明显，接近原系统的频域响应曲线，而传统的 Oustaloup 滤波器效果有较大的偏差。

```
>> b=[1 1]; a=[10,185,288,1]; nb=[1 0]; na=[3.2,2.5,0.7,0];
   w=logspace(-4,4,200); G0=fotf(a,na,b,nb); H=bode(G0,w);
   s=zpk('s'); N=4; w1=1e-3; w2=1e3; b=10; d=9;
   g1=oustafod(0.2,N,w1,w2); g2=oustafod(0.5,N,w1,w2); a1=g1;
   g3=oustafod(0.7,N,w1,w2);
   G1=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1);
   g1=new_fod(0.2,N,w1,w2,b,d); g2=new_fod(0.5,N,w1,w2,b,d);
   g3=new_fod(0.7,N,w1,w2,b,d); bode(g1,'-',a1,'--');
   G2=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1);
   figure; bode(H,'-',G1,'--',G2,':')
```



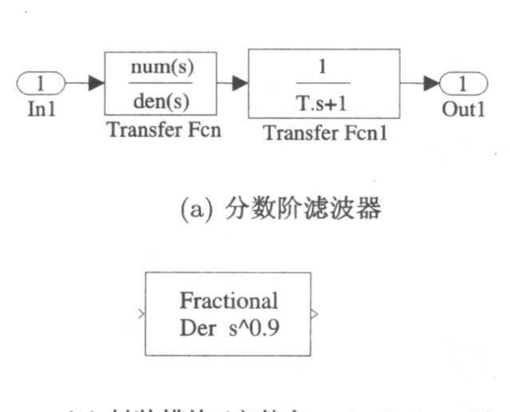
(a) $s^{0.2}$ 微分环节的比较　　(b) 整个系统的 Bode 图比较

图 9-10 两种滤波器的 Bode 图比较

9.3.3 基于 Simulink 框图的分数阶非线性系统仿真方法

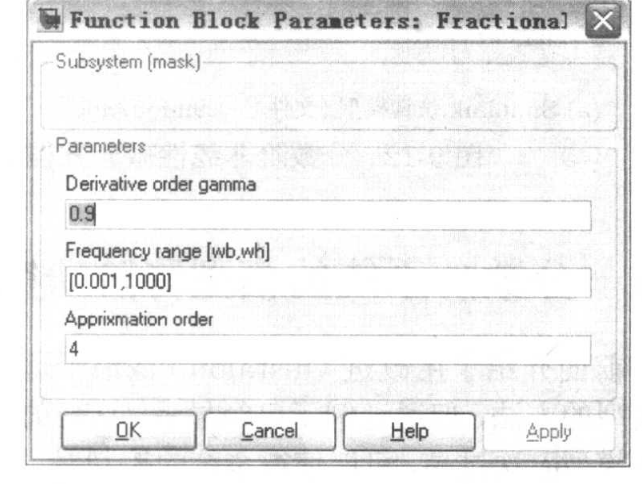
由前面的讨论可见，改进的 Oustaloup 滤波器可以很好地拟合分数阶微分算子，所以可以考虑封装一个 Simulink 模块来表示分数阶微分算子，这样在 Simulink 环境中就可以搭建出任意的非线性分数阶系统模型了。为了避免在仿真时出现代数环，可以在这样定义的微分算子后面串联一个带宽为 $\omega_{\rm h}$ 的低通滤波器，这样构造的 Simulink 模块如图 9-11 (a) 所示。由 Simulink 的封装功能，可以构造出一个分数阶微分算子模块，如图 9-11 (b) 所示。双击该模块，则可以弹出如图 9-11 (c) 所示的参数设置对话框来定义改进的 Oustaloup 滤波器。

```
wb=ww(1); wh=ww(2); G=new_fod(gam,n,wb,wh,10,9);
num=G.num{1}; den=G.den{1}; T=1/wh; str='Fractional\n';
if isnumeric(gam)
   if gam>0, str=[str, 'Der  s^' num2str(gam) ];
   else, str=[str, 'Int  s^{' num2str(gam) '}']; end
else, str=[str, 'Der  s^gam']; end
```



(a) 分数阶滤波器

(b) 封装模块 (文件名：c9mfode.mdl)



(c) 分数阶微分模块参数对话框

图 9-11 分数阶微分算子封装模块构建

在 Simulink 仿真过程中，这样建立的仿真模型可能接近于刚性微分方程，所以应该将算法设置成 ode15s 或 ode23tb，以确保仿真的精度和效率。

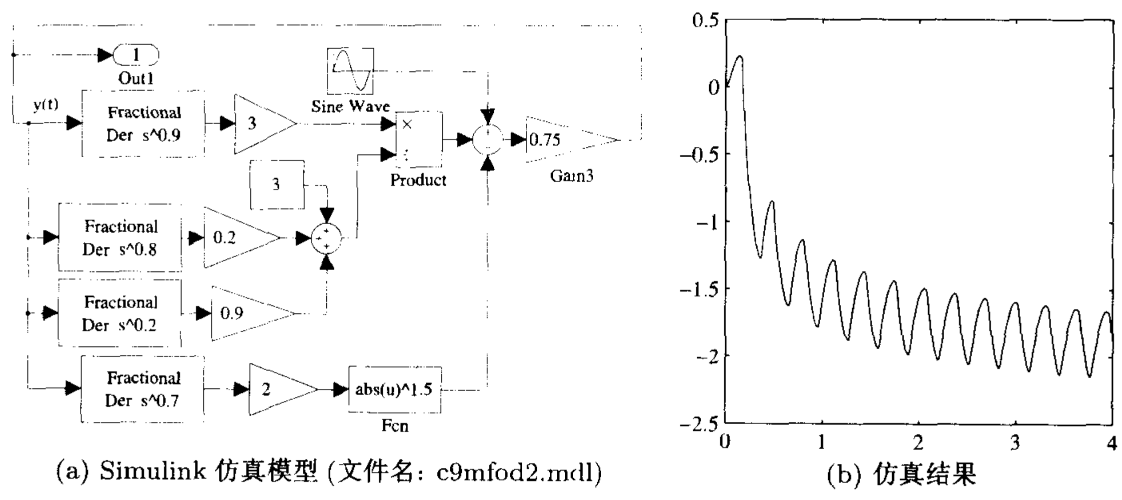
例 9-12 试求解分数阶非线性微分方程。

$$\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}+\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}+\frac{4}{3}y(t)=5\sin(10t)$$

求解 目前一般的分数阶非线性微分方程没有通用的求解方法。对原微分方程稍加处理，则可以推导出 $y(t)$ 的显式表达式

$$y(t)=\frac{3}{4}\left[5\sin(10t)-\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}-\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}\right]$$

根据上面的表达式，可以建立起如图 9-12 (a) 所示的 Simulink 模型。可见，每个分数阶微分算子可以由前面建立的模块表示出来。对该系统进行仿真，则可以得出如图 9-12 (b) 所示的仿真结果。如何对得出结果的正确性加以检验呢？一方面可以考虑用前面介绍的检验方法，选择微分方程求解的控制参数，另一方面可以改进 Oustaloup 滤波器的参数 $\omega_{\rm b}, \omega_{\rm h}, N$。对本系统来说，选择不同参数得出的结果是完全一致的，所以可以认为这样得出的结果是正确的。



(a) Simulink 仿真模型 (文件名：c9mfod2.mdl)　　(b) 仿真结果

图 9-12　分数阶非线性微分方程的仿真模型与仿真结果

9.4　分数阶系统的模型降阶研究

前面介绍了用改进 Oustaloup 滤波器近似分数阶微分算子的形式去逼近线性模型的方法，但通过例子已经显示，这样的拟合将导致非常高阶的整数阶模型。因为模型阶次过高，可能使得系统的控制器设计变得异常困难。所以，用低阶整数阶模型去逼近原模型是有必要的。这和以前介绍的模型降阶研究是密切相关的。

假设降阶模型可以表示为

$$G_{r/m,\tau}(s) = \frac{\beta_1 s^r + \cdots + \beta_r s + \beta_{r+1}}{s^m + \alpha_1 s^{m-1} + \cdots + \alpha_{m-1} s + \alpha_m} \mathrm{e}^{-\tau s} \tag{9-4-1}$$

可以引入一个目标函数来最小化误差信号 $e(t)$ 的 $\mathcal{H}_2$-范数

$$J = \min_{\boldsymbol{\theta}} \left|\left| \widehat{G}(s) - G_{r/m,\tau}(s) \right|\right|_2 \tag{9-4-2}$$

其中 $\boldsymbol{\theta}$ 为需要优化的模型参数，即 $\boldsymbol{\theta} = [\beta_1, \ldots, \beta_r, \alpha_1, \ldots, \alpha_m, \tau]$。为方便求解此目标函数，降阶模型 $G_{r/m,\tau}(s)$ 中的时间延迟项可以用 Padé 近似来逼近，得出 $\widehat{G}_{r/m}(s)$。由模型的次最优降阶算法[10]，可以将目标函数改进为

$$J = \min_{\boldsymbol{\theta}} \left|\left| \widehat{G}(s) - \widehat{G}_{r/m}(s) \right|\right|_2 \tag{9-4-3}$$

直接采用前面介绍的 opt_app() 函数即可得出分数阶模型的最优降阶模型。

例 9-13 重新考虑例 9-10 中给出的分数阶传递函数模型，试选择合适的阶次，求出该模型的降阶模型。

求解 考虑原始的分数阶传递函数，若选择拟合阶次 $N=4$，且频段为 $(10^{-3},10^{3})$，可以用下面的语句直接得出有理近似模型

```
>> N=4; w1=1e-3; w2=1e3; s=tf('s'); g1=new_fod(0.501,N,w1,w2,9,10);
   g2=new_fod(0.42,N,w1,w2,9,10); g3=new_fod(0.798,N,w1,w2,9,10);
   g4=new_fod(0.31,N,w1,w2,9,10); g5=new_fod(0.63,N,w1,w2,9,10);
   G=(-2*g5-4)/(2*s^3*g1+3.8*s^2*g2+2.6*s*g3+2.5*s*g4+1.5);
```

这样得出的模型是高达 58 阶的有理模型

$$G(s)=\frac{\begin{array}{c}-3.3625(s+902.8)(s+856.4)(s+681.8)(s+640.7)(s+588.8)(s+1304)\\ (s+1552)(s+1804)(s+293.6)(s+184.5)(s+146.9)(s+138)\\ (s+126.9)(s+65.71)(s+39.75)(s+31.65)(s+29.74)(s+27.33)\\ (s+15.66)(s+8.564)(s+6.818)(s+6.407)(s+5.888)(s+4.197)\\ (s+1.845)(s+1.469)(s+1.38)(s+1.269)(s+1.196)(s+4455)\\ (s+0.3975)(s+0.3165)(s+0.3085)(s+0.2974)(s+0.2733)\\ (s+0.08564)(s+0.07179)(s+0.06818)(s+0.06407)(s+0.05888)\\ (s+0.01845)(s+0.01593)(s+0.01469)(s+0.0138)(s+0.01269)\\ (s+0.003975)(s+0.00347)(s+0.003165)(s+0.002974)(s+0.002733)\\ (s+0.0008867)(s+0.0006974)(s+0.0005567)(s+0.0004667)(s+0.0003444)\end{array}}{\begin{array}{c}(s+0.003494)(s+0.003163)(s+0.002978)(s+0.002732)(s+0.003974)\\ (s+0.0008869)(s+0.0007)(s+0.0005567)(s+0.0004664)(s+0.0003445)\\ (s+0.01271)(s+0.01379)(s+0.01471)(s+0.01622)(s+0.01845)\\ (s+0.06001)(s+0.06373)(s+0.06836)(s+0.07528)(s+0.08623)\\ (s+0.3196)(s+0.3494)(s+0.4489)(s+1.622)(s+1.641)(s+2.811)\\ (s+5.92)(s+6.274)(s+7.528)(s+8.513)(s+14.51)(s+27.33)\\ (s+29.65)(s+34.94)(s+39.73)(s+67.98)(s+126.9)(s+138)\\ (s+162.2)(s+184.5)(s+315.9)(s+588.8)(s+640.6)(s+752.8)\\ (s+856.4)(s+900)(s+1304)(s+1552)(s+2432)(s+4455)\\ (s^2+0.5861s+0.08621)(s^2+2.592s+1.694)\\ (s^2+0.2218s+0.2263)(s^2+2.592s+3.436)\end{array}}$$

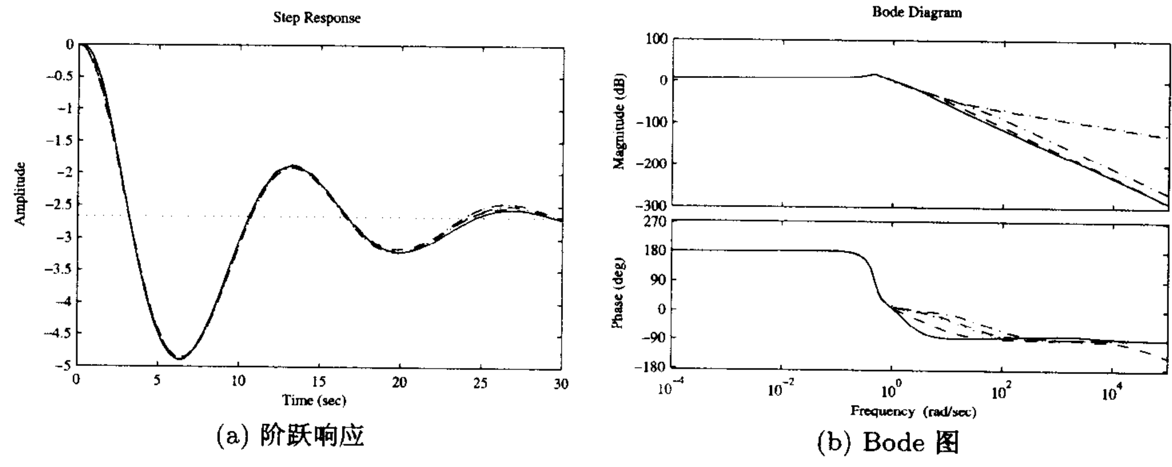
利用改进的 Oustaloup 滤波器算法，选择不同的阶次组合可以得出不同的滤波器模型，拟合误差的 $\mathcal{H}_2$ 范数，在表 9-1 中列出。从表中可见，$G_{2/5}(s)$ 拟合的降阶模型是效果最好的。注意，在这段代码中有时 opt_app() 函数可能需要多次调用来得出收敛的结果。系统的阶跃响应曲线和 Bode 图比较分别如图 9-13 (a)、(b) 所示。

```
>> Gr1=opt_app(G,2,3,0); norm(G-Gr1),
   Gr2=opt_app(G,2,4,0); norm(G-Gr2)
   Gr3=opt_app(G,2,5,0); Gr3=opt_app(G,2,5,0,Gr3); norm(G-Gr3)
```

```
Gr4=opt_app(G,2,6,0); Gr4=opt_app(G,2,6,0,Gr4);
Gr4=opt_app(G,2,6,0,Gr4); norm(G-Gr4)
step(G,'-',Gr1,'--',Gr2,':',Gr3,'-.',Gr4,'--r',30);
figure; bode(G,'-',Gr1,'--',Gr2,':',Gr3,'-.',Gr4,'--r')
```

表 9-1 不同阶次组合下的降阶模型与比较

r	m	降阶模型	误差 $\mathcal{H}_2$
2	3	$\dfrac{0.03147s^2-0.8141s-0.07206}{s^3+0.3168s^2+0.2582s+0.02703}$	0.2286
2	4	$\dfrac{-0.0119s^2-23.21s-2.035}{s^4+28.78s^3+9.242s^2+7.365s+0.7634}$	0.2308
2	5	$\dfrac{-4.932s^2-0.8602s-0.00386}{s^5+5.741s^4+2.794s^3+1.596s^2+0.3134s+0.001448}$	0.1342
2	6	$\dfrac{-2.327\times10^4s^2-4059s-18.21}{s^6+4719s^5+2.709\times10^4s^4+1.318\times10^4s^3+7534s^2+1479s+6.831}$	0.1342



(a) 阶跃响应　　(b) Bode 图

图 9-13 各个降阶模型的比较

9.5 分数阶系统的计算机辅助设计

由前面介绍的分数阶系统分析可见，其分析方法和传统整数阶分析是有区别的。在控制器设计领域也是如此，例如，整数阶系统研究中常用的 PID 控制器可以更一般地用其分数阶版本 $\mathrm{PI}^\lambda\mathrm{D}^\mu$ 控制器来表示[6]

$$G_\mathrm{c}(s)=K_\mathrm{p}+\frac{K_\mathrm{i}}{s^\lambda}+K_\mathrm{d}s^\mu \tag{9-5-1}$$

由图 9-14 中给出的描述可见，横、纵轴分别描述积分器阶次 λ 和微分器阶次 μ，这样传统的整数阶 PI, PD 和 PID 控制器只相当平面上几个特定的点。由于引

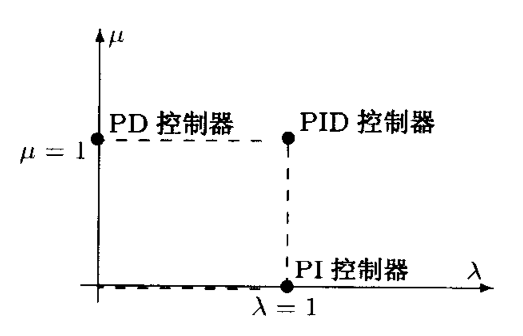


图 9-14 分数阶 PID 控制器示意图

入了两个附加的可调参数 λ 和 μ，控制器参数的调整将更灵活。在某些控制问题中，最好的分数阶 PID 控制器的性能将远远好于最好的整数阶 PID 控制器[11]。

如果从回路成型角度考虑，可见 Bode 图的斜率不再限制为 20 dB/decade 的整数倍，这样，回路传递函数可以根据需要更自由地选择，例如在截止频率附近可以设置得更平些来增加系统的鲁棒性等。

例 9-14 考虑受控对象模型 $G(s)=\dfrac{1}{s^{2.6}+2.2s^{1.5}+2.9s^{1.3}+3.32s^{0.9}+1}$。试先用带有延迟的一阶模型 $G_{\rm p}(s)=\dfrac{k{\rm e}^{-Ls}}{Ts+1}$ 逼近，然后根据该模型设计整数阶 PID 控制器。

求解 由下面的语句可以立即得出一阶近似模型

```
>> N=4; w1=1e-3; w2=1e3; s=tf('s');
   g1=new_fod(0.6,N,w1,w2,9,10); g2=new_fod(0.5,N,w1,w2,9,10);
   g3=new_fod(0.3,N,w1,w2,9,10); g4=new_fod(0.9,N,w1,w2,9,10);
   G=1/(s^2*g1+2.2*s*g2+2.9*s*g3+3.32*g4+1); Gr=opt_app(G,0,1,1)
```

这时得出的模型为 $G_{\rm r}(s)=\dfrac{0.1702}{s+0.1702}{\rm e}^{-0.612s}$。其实由这样的模型，可以通过各种不同的算法来设计整数阶 PID 控制器[12,13]，例如常用Wang-Juang-Chan 算法[14]

$$K_{\rm p}=\frac{(0.7303+0.5307T/L)(T+0.5L)}{K(T+L)},\ T_{\rm i}=T+0.5L,\ T_{\rm d}=\frac{0.5LT}{T+0.5L}\tag{9-5-2}$$

根据降阶模型参数可以容易地设计出整数阶 PID 控制器

```
>> K=0.1702/0.1702; T=1/0.1702; L=0.612;
   Ti=T+0.5*L; Kp=(0.7303+0.5307*T/L)*Ti/(K*(T+L));
   Td=(0.5*L*T)/(T+0.5*L); Gc=Kp*(1+1/Ti/s+Td*s),
```

控制器为 $G_{\rm c}(s)=4.7960\left(1+\dfrac{1}{5.6315s}+0.3076s\right)=\dfrac{1.614s^2+5.55s+0.8979}{s}$。在该控制器作用下，可以由下面语句绘制出闭环系统的阶跃响应曲线，如图 9-15 所示。可见，这样得出的控制效果是理想的。另外，值得指出的是，这里的阶跃响应曲线分别为 Grünwald-Letnikov 算法直接计算的方法和高阶整数阶近似直接计算的方法得出的，从结果看，两种方法得出的结果基本一致。

```
>> Gcf=fotf(1,1,[1.614 5.55 0.8979],[2,1,0]);
   a=[1 2.2 2.9 3.32 1]; an=[2.6,1.5,1.3 0.9 0];
```

```
G0=fotf(a,an,1,0); GG=feedback(Gcf*G0,1); t=0:0.005:15;
step(feedback(G*Gc,1),t); hold on, step(feedback(G0*Gcf,1),t);
```

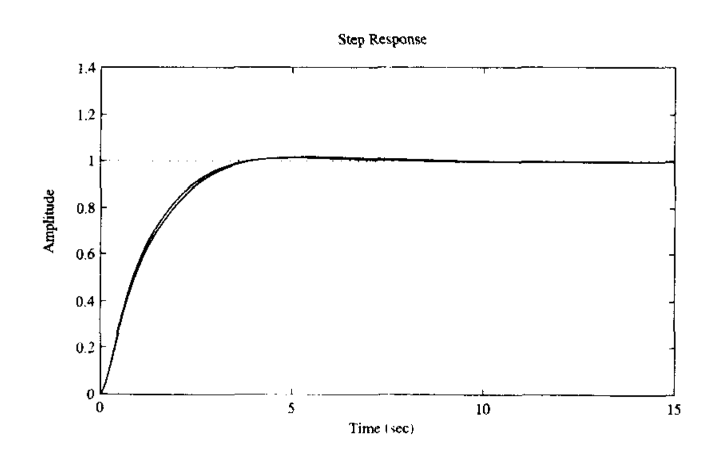


图 9-15 分数阶受控对象的整数阶 PID 控制阶跃响应曲线

9.6 习题与思考题

1 给定信号 $f(t)=\mathrm{e}^{-3t}\sin(t+\pi/3)+t^2+3t+2$，试利用定义计算出该函数的 0.2 阶微分信号及 0.7 阶积分信号，并将结果信号用曲线表示出来。

2 对下面给出的函数，试用三维表面图的形式表示其微分函数与阶次、时间之间的关系，并比较当阶次为 1 时结果是否与整数阶微分一致。

① $f(t)=\sqrt{t\sin t\sqrt{1-\mathrm{e}^t}}$，② $y(t)=\sqrt{\dfrac{(t-1)(t-2)}{(t-3)(t-4)}}$

3 本章给出了多种分数阶微分的定义，但只给出了其中的 Grünwald-Letnikov 定义的 MATLAB 实现，试利用数值积分的方法由其他定义计算某给定函数的各阶导数，并与 Grünwald-Letnikov 定义进行比较。

4 假设分数阶线性微分方程为 $0.8\mathscr{D}_t^{2.2}y(t)+0.5\mathscr{D}_t^{0.9}y(t)+y(t)=1$，其初始条件为 $y(0)=y'(0)=y''(0)=0$。试求出该方程的数值解。如果将分数阶次 2.2 近似成 2，0.9 近似成 1，则可以得出整数阶近似模型。试比较分数阶系统和整数阶系统的相似性。

5 假设已知典型反馈系统结构，求闭环分数阶模型，其中第一个系统为整数阶模型，试用分数阶的方法构造闭环模型，并和整数阶得出的模型进行比较。

① $G(s)=\dfrac{211.87s+317.64}{(s+20)(s+94.34)(s+0.17)}$，$G_\mathrm{c}(s)=\dfrac{169.6s+400}{s(s+4)}$，$H(s)=\dfrac{1}{0.01s+1}$

② $G(s)=\dfrac{s^{0.4}+5}{s^{3.1}+2.8s^{2.2}+1.5s^{0.8}+4}$，$G_\mathrm{c}(s)=3+2.5s^{-0.5}+1.4s^{0.8}$，$H(s)=1$

6 本章介绍了类的构造和重载，并建立起分数阶传递函数对象类，定义了各个算符和诸如 step() 这样的函数，试仿照控制系统工具箱中的函数扩展分数阶传递函数类的重载函数。

7 分数阶系统稳定性的判定是一个较复杂的问题。对一个简单的分数阶模型 $G(s)=\dfrac{5}{s^\alpha-1}$，试对不同的 α 值进行试凑，观察是否存在使系统阶跃响应稳定的 α 值。

8 考虑下面给出的分数阶模型

① $G(s)=\dfrac{5}{s^{2.3}+1.3s^{0.9}+1.25}$，② $G(s)=\dfrac{5s^{0.6}+2}{s^{3.3}+3.1s^{2.6}+2.89s^{1.9}+2.5s^{1.4}+1.2}$

试得出近似的低阶整数阶模型，并比较得出的近似模型和原分数阶模型在频域响应和时域响应方面的性质。

9 考虑分数阶受控对象模型 $G(s)=\dfrac{1}{s^{2.6}+2.2s^{1.5}+2.9s^{1.3}+3.32s^{0.9}+1}$，一个整数阶 PID 模型可以表示为 $G_c(s)=\dfrac{1.614s^2+5.55s+0.8979}{s}$，试得出闭环的分数阶模型，并观察该闭环系统的阶跃响应曲线。

10 试求解下面 Mittag-Leffler 函数，绘制出相应的曲线并验证式 (9-2-7)。

① $\mathscr{E}_{1,1}(z)$，② $\mathscr{E}_{2,1}(z)$，③ $\mathscr{E}_{1,2}(z)$，④ $\mathscr{E}_{2,2}(z)$

11 利用本章给出的 Mittag-Leffler 函数代码，用数值方法验证下面几个等式。

① $\mathscr{E}_{\alpha,\beta}(x)+\mathscr{E}_{\alpha,\beta}(-x)=2\mathscr{E}_{\alpha,\beta}(x^2)$，② $\mathscr{E}_{\alpha,\beta}(x)-\mathscr{E}_{\alpha,\beta}(-x)=2x\mathscr{E}_{\alpha,\alpha+\beta}(x^2)$

③ $\mathscr{E}_{\alpha,\beta}(x)=\dfrac{1}{\Gamma(\beta)}+\mathscr{E}_{\alpha,\alpha+\beta}(x)$，④ $\mathscr{E}_{\alpha,\beta}(x)=\beta\mathscr{E}_{\alpha,\beta+1}(x)+\alpha x\dfrac{\mathrm{d}}{\mathrm{d}x}\mathscr{E}_{\alpha,\beta+1}(x)$

12 式 (9-2-11) 中给出了二阶特殊系统的 Mittag-Leffler 函数求解式子，试仿照此构造三阶系统的解法，并编写出求解此类问题的 MATLAB 程序。对受控对象模型 $G(s)=\dfrac{4s^{0.4}+3}{s^{2.3}+2s^{1.6}+1.8s^{0.8}+4}$ 求解其阶跃响应，并和基于 Grünwald-Letnikov 算法得出的结果相比较。

13 考虑分数阶线性微分方程 $\mathscr{D}x(t)+\left(\dfrac{9}{1+2\lambda}\right)^\alpha\mathscr{D}^\alpha x(t)+x(t)=1,\ 0<\alpha<1$，其中 $\lambda=0.5,\ \alpha=0.25$。试用数值方法求解该微分方程。

14 对分数阶微分算子 $s^{0.7}$，使用 Oustaloup 滤波器和改进的滤波器对其近似，试找出能够较好地逼近效果分数阶微分的 N 阶次。

15 从前面给出的演示可见，改进的滤波器效果明显优于 Oustaloup 滤波器，试通过仿真实验给出其参数 b,d 的选择方法。

16 试构造出 Simulink 仿真框图，求解下面的分数阶非线性微分方程。

$$\mathscr{D}^2x(t)+\mathscr{D}^{1.455}x(t)+\left[\mathscr{D}^{0.555}x(t)\right]^2+x^3(t)=f(t)$$

17 考虑分数阶受控对象模型 $G(s)=\dfrac{5s^{0.6}+2}{s^{3.3}+3.1s^{2.6}+2.89s^{1.9}+2.5s^{1.4}+1.2}$，试设计一个整数阶 PID 控制器，观察控制效果。

18 试为分数阶受控对象 $G=\dfrac{b}{as^{0.7}+1}$ 设计一个 $\mathcal{H}_\infty$ 控制器，使得该控制器在参数 $a\in(0.2,5)$ 和 $b\in(0.2,1.5)$ 的变化下仍然能有较好的控制效果。

参考文献

[1] Vinagre B M, Chen Y Q. Fractional calculus applications in automatic control and robotics [M]. Las Vegas: 41st IEEE CDC, Tutorial workshop 2, 2002

[2] Podlubny I. Fractional differential equations [M]. San Diago: Academic Press, 1999

[3] 薛定宇，陈阳泉. 高等应用数学问题的 MATLAB 求解 [M]. 北京: 清华大学出版社, 2004

[4] Hilfer R. Applications of fractional calculus in physics [M]. Singapore: World Scientific, 2000

[5] Petráš I, Podlubny I, O'Leary P. Analogue realization of fractional order controllers [M]. Fakulta BERG, TU Košice, 2002

[6] Podlubny I. Fractional-order systems and $\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}$-controllers [J]. IEEE Transactions on Automatic Control, 1999, 44(1):208~214

[7] Chen Y Q, Vinagre B M. A new IIR-type digital fractional order differentiator [J]. Signal Processing, 2003, 83:2359~2365

[8] Oustaloup A, Levron F, Mathieu B, et al. Frequency-band complex noninteger differentiator: characterization and synthesis [J]. IEEE Transactions on Circuit and Systems-I: Fundamental Theory and Applications, 2000, TCS-47(1):25~39

[9] Xue D, Zhao C N, Chen Y Q. A modified approximation method of fractional order system [A]. Proceedings of IEEE Conference on Mechatronics and Automation [C]. Luoyang, China, 2006, 1043~1048

[10] Xue D, Atherton D P. A suboptimal reduction algorithm for linear systems with a time delay [J]. International Journal of Control, 1994, 60(2):181~196

[11] Xue D, Zhao C N, Chen Y Q. Fractional order PID control of a DC-motor with elastic shaft: a case study [A]. Proceedings of American Control Conference [C]. Minneapolis, Minnesota, 2006, 3182~3187

[12] O'Dwyer A. PI and PID controller tuning rules for time delay processes: a summary. Part 1 & 2 [A]. Proceedings of the Irish Signals and Systems Conference [C]. 1999

[13] O'Dwyer A. Handbook of PI and PID controller tuning rules [M]. London: Imperial College Press, 2003

[14] Wang F S, Juang W S, Chan C T. Optimal tuning of PID controllers for single and cascade control loops [J]. Chemical Engineering Communications, 1995, 132:15~34

函数名索引

本书涉及大量的 MATLAB 函数与作者编写的 MATLAB 程序，为方便查阅与参考，这里给出重要的 MATLAB 函数调用语句的索引，其中黑体字页码表示函数定义和调用格式页。

专业术语索引